Employee Group Benefits Rating Engine

Requirements Summary and Design Specification

The perception that the rating engine specification and design would take a substantial amount of time caused InsWeb to begin that part of the Employee Group Benefits application first. Version 1.0 of this document is the result of that preliminary effort. As other aspects of the application are specified and designed, it is recognized that this document will undergo substantial revision. It is important, however, that other development team members understand the basics of the rating engine, since many parts of the application must interact with it. Therefore, this document is being published and distributed in this preliminary form.

Version 1.0

Table of Contents

1. INTRODUCTION

1.1 OVERVIEW

InsWeb intends to include Employee Group Benefits with its other insurance offerings. Because of its complexity, development of the Employee Group Benefits Rating Engine (EGBRE) design is regarded to be the most time consuming portion of the software development project, so work on it precedes other portions of the project.

No formal requirements specification currently exists for the planned Employee Group Benefits application, so this document begins with a summary of the requirements specifications as they are currently known. Requirements and design regarding other portions are included only to the extent necessary to justify design of the rating engine.

1.2 PURPOSE

Beyond its use in designing and implementing the EGBRE, the purpose of this document is to solicit feedback from others to ensure that it is consistent with InsWeb and insurance industry conventions.

1.3 CONTEXT

Initial focus is on the group medical portion of Employee Group Benefits, but as details unfold regarding group dental, long and short-term disability, long-term care and vision products, and group life/accident/disability this design framework will be modified to include these other portions.

It is important that this work and its results be consistent with other work existing or in progress at InsWeb. Of concern are:

- Front end look and feel
- Web page delivery architecture
- Object model
- Existing software
- Database entities

Where rating factor tables and algorithms are similar to those in existing InsWeb systems, feasibility of reusing existing software has been reviewed.

To the extent that other insurance applications are compatible with Employee Group Benefits insurance types, general object relationships and database structures are designed so that they can be used in other InsWeb applications.

This document does not include specifications of InsWeb project management, object architecture, data validation and error reporting, details of programs which call the rating engine, or billing for rating services. They are described elsewhere.

2. REQUIREMENTS SUMMARY

2.1 FUNDAMENTAL CONCEPTS

This section is intended to provide an overview of the context in which a rating takes place. This discussion is intended to bridge the gap between insurance industry terminology and application development terminology. Note that these concepts are discussed in abstract terms emphasizing data form rather than data content. The result of this discussion will be to define a candidate set of processing and data structures. Once these structures are defined, it will be necessary to review the requirements to demonstrate that all can be met in a manner that promotes high performance and straightforward application development.

2.1.1 FACTORS

The term factor is used in the insurance industry in a variety of ways. This term is discussed first to reduce potential confusion.

2.1.1.1 Consumer (or employee) Factor

An attribute of a consumer (e.g. location) or employee (e.g. age). Each consumer or employee factor has a value (location = San Mateo, or location = 94402, or location = Northern California, age = 39, etc.) which will be called consumer factor value or employee factor value.

2.1.1.2 Rating Factor

Consumer and employee factors and their values are used to determine rating factors, usually with a table lookup. Where a table calls for use of a particular consumer factor, the consumer factor value is used as an index to the table. The rating factor value found in the table is returned to be used in the calculation.

2.1.1.3 Product Factor

The various coverages of a policy are grouped into one or more segments (groups of coverages) for calculating and reporting purposes. Each segment is divided into one or more product factors, each of which frequently correspond to a single consumer or employee factor.

In this simple case, the single consumer or employee factor value is used for a table lookup which returns a single rating factor value. In more complex cases the product factor corresponds to multiple consumer and employee factors, along with coverage options which are used for a multiple parameter table lookup to return a single rating factor value.

In the most complex situations a single product factor can correspond to several sets of consumer and employee factors and coverage options, each of which corresponds to a table lookup. The several rating factors returned can be multiplied together, one can be the exponent of the other, or one can be used as one of the lookup values for the subsequent table lookup.

After all product factors are determined, they are multiplied together and then by a product factor base value to determine the segment value. Usually one of the product factors will have units of dollars and the others will be unitless. The final segment value will always be rounded to dollars and whole cents.

2.1.2 PRODUCT

The insurance product is the primary product of an insurance company and one of two central data objects of concern to the rating engine. A product consists of:

2.1.2.1 Identifying attributes

To uniquely identify an insurance product, it will be necessary to know some or all of the following attributes.

- Product Name
- Carrier
- Major geographic area
- Product type
- Group size, range
- Network
- Unique Product ID. For various companies and products, different combinations of other identifying attributes may be used to uniquely identify a product. The product ID will be used internally for unique identification for the sake of consistency. No application consumers or carriers need to be aware of this attribute.

2.1.2.2 Coverage and Options

Coverages are attributes of products which define that which is actually being covered, and therefore its value. The consumer may exercise control to fine tune the product to best meet the consumer's requirements by selecting values for coverages, which are called options.

- Options are the set of allowed values for a given coverage. An option is the selected value for a coverage.

- Not all combinations of selected options will be allowed for a given product. For simplicity, combinations are limited by the carrier to three basic policies allowing only a restricted set of options to the consumer. Coverage for which consumers will be allowed to choose among multiple options beyond the basic three policies are:
  - Deductible
  - Coinsurance
  - Stop-loss
  - Various copayments
  - Group size
  - State or major geographic area
  - Network

- A list of all currently known coverages is included in the Product Data Entry Data Interface section below.

2.1.2.3 Policy

Once an option is chosen for each product coverage, the combination is known as a policy. This is the specific product being considered for purchase by a consumer and for which a rating is requested.

2.1.2.4 Segment

Not all parts of a policy's coverage are affected by consumer and employee factors in the same way. For example, some parts may be affected by age, sex, and geographic area factors, while others may depend only on headcount. Therefore, the product may be divided into parts, called segments, with each segment having factors applied to it in a way different than factors are applied to other segments. Each segment has a base value (which comprises the calculation starting point).

Each segment is composed of one or more product factors. Once calculated, the product factors are multiplied together, then multiplied by the segment base value to yield the segment value. Segment values are added together to provide the employee or consumer rating.

2.1.3 CONSUMER

The current or prospective purchaser of insurance. For the Employee Benefits application a consumer is a business with multiple employees. Each employee who elects to receive coverage represents one or more insured persons called members. A consumer has:

2.1.3.1 Consumer Factors:

Consumer factors are attributes (e.g. geographic area of employer) of the consumer used to adjust the cost of the insurance product (policy rating) being offered.

- Consumer factors have names which identify them, and numeric values.
- Once consumer and employee factors have been applied to adjust the policy rating, it is called the consumer's rating or simply, the rating.
- A list of all currently known consumer factors is included in the Product Data Entry Data Interface section below.

2.1.3.2 Employee

Any employee of the consumer (employer) seeking coverage for any of the insurance types to be covered. Each employee is a member of the group associated with each insurance type product if they participate in that type of insurance.

2.1.3.3 Employee Factors:

Employee factors are attributes (e.g. age) of the employee used to adjust the cost of the insurance product (policy rating) being offered.

- Employee factors have names which identify them, and numeric values.
- Once employee factors have been applied to adjust the policy rating, it is called the employee's rating.
- A list of all currently known employee factors is included in the Product Data Entry Data Interface section below.

2.1.3.4 Risk Profile

A risk profile for a consumer is the set of all consumer factors and employee factors for that consumer. The risk profile information is used along with the policy and rating table information to produce a rating.

2.1.4 RATING

A dollar amount per month to be charged to the employer. The rating is calculated using a formula specific to the Product. The framework in which a specific formula is calculated is described as follows:

- The rate is equal to a set of segment values added together.
- Each segment value is comprised of a leading numeric factor multiplied by zero or more additional numeric product factors.
- Each product factor is a numeric value determined from one or more tables, each of which is indexed by various consumer and employee factors and product coverage options. If there is more than one table for the product factor the product factor also must include information about how to relate those tables (i.e. multiply them together, use one as the exponent for the other, etc.)

2.2 INTERFACES

The Employee Group Benefits application will be comprised of seven modules (see diagram). Three of these interact extensively with the EGBRE directly by requesting one or more ratings to be performed or indirectly by providing data critical to rating calculations for later use.. Requirements and design of the other six modules will only be discussed to the extent to which they affect the EGBRE design.

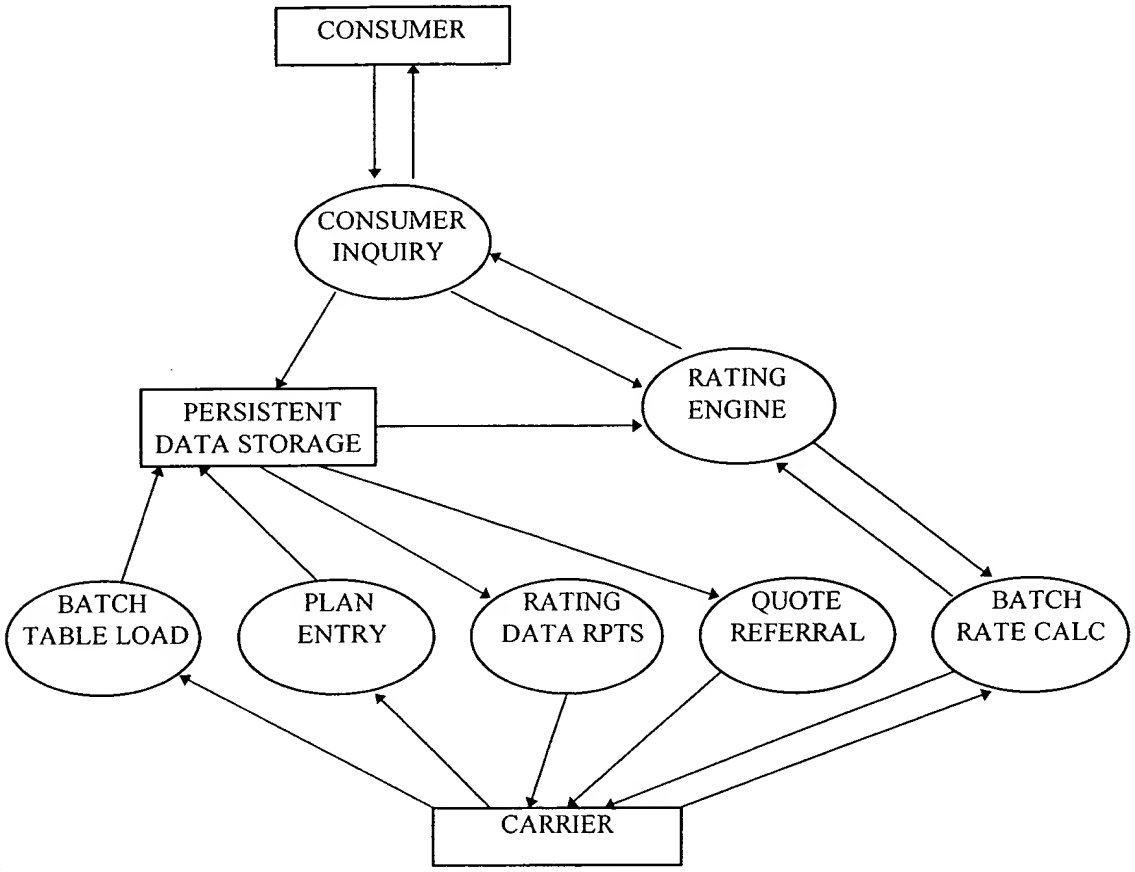

fig 1. Employee Group Benefits Functions and Interfaces

2.2.1 CONSUMER INQUIRY INTERFACE

2.2.1.1 Functionality

The consumer inquiry interface will collect certain information from consumers, known as the risk profile. This information will be made available to the EGBRE in addition to a list of the consumer's requested policies.

- Consumers must be able to enter employer and census information (also known as a risk profile) over several sessions, modifying and adding to it as desired. Thus, there must be a means of persistent data storage for consumer information.

- Consumers must be able to return days, even weeks later (up to one year) to obtain additional ratings based on the same or on a modified risk profile.

- Consumers must be able to specify option values for a set of coverages.
- Consumers must be able to request ratings for one or a group of policies and to repeat the process as desired.
- For each policy, the EGBRE must be able to return total rating, rating by segment, rating by employee, and rating by employee by segment.
- The EGBRE must be able to provide meaningful error information regarding missing or inconsistent data from the consumer's risk profile.
- The EGBRE must be able to log detailed diagnostic data used to make a rating. It must be possible for an administrator to turn this feature on and off without restarting the application.
- The EGBRE must be able to log summary information provided to the consumer inquiry interface for tracking purposes. This feature must be optional and depend on a request from the consumer inquiry interface.
- Consumer location will be presented in Zip5 format. It must be possible for the EGBRE to determine which one, two, three, four, or five digit zip code the Consumer is within.

2.2.1.2 Data

The following data associated with the consumer inquiry interface will be made available to the EGBRE.

Data types and field lengths are yet to be determined. They will depend on consumer inquiry and product data entry interface requirements, data standards used in the insurance industry, and other InsWeb applications and standards.

2.2.1.2.1 Direct

This data will have been collected within the same session during which the rating is requested.

Consumer identification, which must be an internally used ConsumerID and which may be associated with an anonymous identifier and/or actual employer identification. The internal identifier will be sufficient for the rating engine to correctly associate all employer factors and census data.

Policy identifying information for one or more policies as selected by the consumer. There may be multiple carriers included, multiple products for a carrier, and multiple policies for a product.

- CarrierID - As selected by the consumer.
- ProductID - As selected by the consumer.
- PolicyNo - As selected by the consumer.
- Consumer Option (CoverageID, OptionNo) - While many of the options are set by the policy, some are left for the consumer's selection. If the policy has an option it is used. If not, if the consumer has selected an option it is used. If neither selection has been made, then the coverages default option will be used. The consumer inquiry interface must determine which policies and which options best meet the consumer's criteria, then make that information available to the EGBRE.
- Rating Date/Time

Note that a single request to the EGBRE will be for a single consumer, but may be for one or more policies.

2.2.1.2.2 Indirect

This data may have been collected in the same session in which a rating is requested, or it may have been gathered during one or more previous sessions.

- The rating engine must allow new consumer and employee factors and new units for existing factors to be added without modifying the rating engine programming.

Therefore, this list of consumer and employee census factors is only suggestive of the factors necessary to perform a rating.

- Consumer Factors - All consumer attributes (employer) used by the various carriers which affect the policy's rating will be collected as consumer factors. Not all consumer factors are necessary to calculate a rating for every policy. They are expected to include:

    - Geographic area - The consumer inquiry interface must specify the units (e.g. zip, county, state). They must be in the same units in which the policy is described (this will not generally be true) or there must be a means of determining whether the consumer's location is within the policy's areas (e.g. conversion or inclusion tables)

    - Rating Date - The date for which the rating is to be effective. It is expected that this will frequently, but not always be the system's current date. This date is used to calculate the trend factor, which depends on the number of months since the policy became effective. It must also be included with logged data for later review. The rating date must also be used to determine which versions of tables are in effect at the time of the rating.

    - Group Size - total number of employees seeking coverage of any type.

    - Industry Factor - Standard Industry Code or other designation.

- Pre-existing insurance factor (medical)
- Pre-announced factor (dental)
- Employee turnover rate (dental)
- Participation percentage (dental)

- Employee Census Factors - This information is to be included for each employee who is to be included in any type of coverage.
  - Employee age and sex
  - Spouse age and sex
  - Number of children
  - Family status (may be inferred from existence of spouse and children). Single, married, single with children, married with children. Note that the application requesting a rating must be able to provide this information according to several different tier structures.
  - Healthy lifestyle (non-smoker) Y/N
  - Hourly or Salaried (dental)
  - Employee salary (short-term disability; if coverage is percentage of salary, not if the coverage is a fixed cap)
  - Medical Group
  - Dental Group
  - Life Group
  - Short-term disability Group
  - Long-term disability Group
  - Long-term care Group
  - Vision Group

2.2.1.2.3 Output

The EGBRE will provide the following data:

- Total rating
- Rating by employee
- Rating by segment
- Rating by employee by segment

2.2.2 PRODUCT DATA ENTRY

2.2.2.1 Functionality

Information necessary to characterize each policy for rating purposes must be made available to the EGBRE. This must be in a persistent form, continuously available to the EGBRE.

- EGBRE must be able to identify the different segments of a policy and be able to perform separate ratings on each segment (as requested).
- EGBRE must be able to identify which product factors belong to a segment and to multiply them together to calculate the segments value.
- EGBRE must be able determine the value of each product factor by using product option values in combination with consumer and employee factor values to perform rating table lookups.
- EGBRE must be able to evaluate product factors which depend on up to five total product option values, consumer factor values, and employee factor values.
- EGBRE must be able to accept controlling parameters telling it how to arithmetically combine various product factors. It must be possible to add, multiply, and exponentiate product factors. It must also be possible to use one retrieved product factor as the basis for retrieving another product factor.
- Rating tables used must include an effective date range indicating when they are to be used by EGBRE. As a table "times out", EGBRE must use the newer version of the table without explicit intervention.
- It must be possible to specify a minimum value and a maximum value for a given product factor (known as bracketing, which is done for the total census factor for legislatively mandated products).
- Product tables will use the leading one, two, three, four, or five digits of the Zip code. It must be possible for the EGBRE to determine which table entry corresponds to which five digit consumer zip. Please note that this issue is not yet completely resolved. Further input from the consumer inquiry interface design effort will be necessary.

2.2.2.2 Data

Data regarding products comes from both the Product Data Entry module and the Batch Rating Data Upload module. Because the rating engine retrieves this data from persistent storage it does not interact directly with either module. The data to be provided is detailed here and not repeated in the Batch Rating Data Upload section.

Data types and field lengths are yet to be determined. They will depend on interface requirements, data standards used in the insurance industry, and other InsWeb applications.

2.2.2.2.1 Direct

There will be no data passed directly to the EGBRE from the Product Data Entry interface.

2.2.2.2.2 Indirect

This data will have been collected during one or more previous sessions. It is stored by the Product Data Entry Interface (or Batch Rating Data Upload) in persistent storage.

- The rating engine must allow new coverages and options and new units for existing options to be added without modifying the rating engine programming.

Therefore, this list of coverages is only suggestive of the coverages necessary to perform a rating. Many of the options will have dollar values, some will have "Y" or "N" values, and some will have arbitrary strings for values.

- Product type
- Stop-loss
- Yearly deductible, individual
- Yearly deductible, family
- Yearly coinsurance
- Yearly out of pocket maximum
- Physician PCP services copay
- Specialist services copay
- Chiropractic services copay
- Inpatient hospital copay
- Outpatient hospital emergency room copay
- Outpatient hospital surgery copay
- Prescription drugs generic copay
- Prescription drugs brand copay
- Maternity
- Mental health
- Substance abuse
- Durable medical equipment
- First dollar accident coverage
- Transplant coverage
- Infertility drugs
- Home health
- Skilled nursing facility
- Preventive coverages
- Hospice coverages
- Network
- Minimum group size (may determine eligibility)
- Maximum group size (may determine eligibility)
- Geographic area

2.2.3 REFERRAL INTERFACE

Depending on consumer request and possibly other factors, the Employee Group Benefits application transmits consumer supplied and other requested information to the insurance company, either immediately or as part of a later batch process. Summary logging data stored by the EGBRE may be used for this purpose.

2.2.3.1 Data (output)

This data will include:

- Consumer identification
- Plan design identification
- Total rating
- Rating by employee
- Rating by segment
- Rating by employee by segment

2.2.4 RATING DATA REPORTING

Rating Data Reporting will enable insurance product maintainers to confirm that product data has been correctly entered and is complete. Data presented will be of two types:

- Tabular presentation of raw option and factor data driving rate calculations.
- Formatted reports representing insurance product structure, options, factor calculation formulae, etc.

2.2.4.1 Data

There will be no data interface between the EGBRE and the Rating Data Reporting module.

2.2.5 BATCH RATING TABLE UPLOAD

Focus will be on determining the complex data mapping necessary to automatically convert data from legacy formats to InsWeb formats and a metadata framework for simplifying the process of adding new formats as new insurance companies see the light.

2.2.5.1 Data

The requirements for data interface between the Batch Rating Table Upload module and the Product Data Entry module is identical and described in the product data entry module above.

2.2.6 BATCH RATING CALCULATION

The batch rating calculation module calculates the ratings for many combinations of options and factors included in batch file. It will be used for internal testing and to convince insurance companies that the rating engine is reliable.

2.2.6.1 Data

The requirements for data interface between the batch rating calculation module and the consumer inquiry interface module is identical and described in the consumer inquiry interface module above.

2.3 DATA FLOWS

2.3.1 BACKGROUND DATA SETUP

This rarely changed data will be entered by an InsWeb administrator. Data includes product types, insurance companies, coverages, options, and networks. This data is not specific to any product or consumer. Data flow is trivial, since it can be entered one record at a time. This data is used primarily for verification and for providing full names for data display.

2.3.2 TABLE ENTRY

This rarely changed data comprises the raw numbers used to designate how a consumer's attributes (geographic area, age distribution) affect rates. Some tables will be used for multiple products and even multiple insurance companies, but some tables will be specific to products and even policies.

It is expected that most table data entry will be via batch import from legacy systems. Batch entry will be concerned with matching legacy key information with InsWeb key information rather than with complex data flows.

The online table maintenance data flow will be trivial, consisting of identifying the table and record to be maintained, then entering or modifying the table entry.

2.3.3 PRODUCT SETUP

This data describes a product, including all parameters necessary to define the rate calculation formulae. This data will be entered by someone expert in insurance product structures who will be carrier's staff or dedicated InsWeb staff (or 3rd party contractors). Steps are:

1. Enter product header information.
2. Enter product design header information.
3. Select option values for each coverage included to create a policy.
4. Enter header information for each segment of the policy.
5. Enter initial coefficient for each segment.
6. Designate which product factors apply to each segment.
7. Designate which table or calculation method is to be used to determine the product factor value.

2.3.4 CONSUMER DATA ENTRY

Consumer will provide sufficient information that a rate can be calculated. Consumer will:

1. Enter consumer identifying information. If the user is not required to supply actual identification a pseudonym supplied by the user or supplied by the system will be used.
2. Enter preliminary information to guide for which factors to prompt (e.g. product type or insurance company of interest, geographic area)
3. Enter factor values. In some cases this may involve moving to a different page to determine a complex value (e.g. census information) and returning.
4. Request and review the rate.
5. Modify factor values, policy, product, insurance company, or other items, without having to reenter unchanged data.
6. Request and review new rate.
7. Save values, enter additional data, and request referral to carrier. Data is archived.

2.3.5 REFERRAL

On a regular basis (perhaps real time) information regarding consumers who have retrieved rate information will be downloaded to insurance companies. Possible scenarios include:

- Once a consumer has seen a quote and has committed to a carrier, consumer, policy, and quote information will be passed to the carrier.

- Monthly statistical information including hits and quotes will be passed to the carrier.

2.3.6 RATING CALCULATION

Before performing the rating calculation, an evaluation is performed to determine if the consumer is eligible for the policy. This is done by a separate process which compares consumer factors with policy criteria to determine if the product is available to the consumer (depending on interface design, it may not be possible for the consumer to choose a policy for which he is not eligible).

The EGBRE receives policy and consumer data, determines the product's details and calculation formulae, and retrieves table values. EGBRE returns a rating (dollar amount per month per employee per segment).

Information retrieved from the data store includes:

- Plan design segment headers.
- Plan design factors (blend and expenses).
- Plan design calculation parameters (which algorithm to use under which circumstances).
- Consumer factors.
- Table entries keyed on the above.

2.4 ALGORITHMS

2.4.1 RATING EVALUATION LOGIC

The EGBRE does not perform or participate in the rating evaluation logic, but this information is included here for convenience.

For the submitted Policy ID and Consumer (quote) ID.
Determine that all required data has been entered.
Verify that geographic area is defined in, or convertible to, correct units (county, zip, or other as required by policy).
Determine that the Consumer qualifies for the Policy
Verify that consumer group size (participating employees) is >= Policy group minimum size
Verify that consumer group size (participating employees) is <= Policy group maximum size
Verify that consumer location (employer, not employee) falls within the Policy's availability area.

2.4.2 RATING ENGINE LOGIC

2.4.2.1 Basic Algorithm

The following outlines not only the manner in which data is added and multiplied together, but how the rating engine identifies all the data necessary for the rating.

2.4.2.1.1 Inputs

The following data is provided directly to the Rating Engine.

- ConsumerID
- CarrierID
- ProductID
- PolicyNo
- Rating Type (One of "Total rating", "rating by segment", "rating by employee", or "rating by employee by segment")

The CarrierID, ProductID, and PolicyNo are necessary to uniquely identify the Policy to be rated. The ConsumerID is necessary to uniquely identify the Consumer to be rated.

2.4.2.1.2 Outputs

The following data is returned by the Rating Engine to the calling program.
For the first Policy Segment and each Segment which is identified as separable:

- Rating
- Each SegmentNo, SegmentRating (if Segment rating was requested)
- Each EmployeeID, Rating (if Employee rating was requested)
- Each EmployeeID, SegmentID, EmployeeSegmentRating (if Employee Segment rating was requested)

2.4.2.1.3 Total Rating and Rating by Segment

Set Rating = 0
For each Segment for the Policy
    Set SegmentRating = Segment.BaseValue
    For each ProductFactor for the Segment
        If any RatingTableKey.FactorType of the RatingTable of the ProductFactor = "E" and the Rating by Employee option was specified:
            Set EmployeeRating = 0
            For each employee who is participating in the Product's Group (for whom exists an EmployeeFactor named the same as the Product.InsuranceType equal to "Y")
                Calculate the ProductFactor Value as described in the next section.
                Set EmployeeRating = EmployeeRating + Value
            Set ProductFactor Value = EmployeeRating
        Else
            Calculate the ProductFactor Value as described in the next section.
        EndElse
        Set SegmentRating = SegmentRating * ProductFactor.BaseValue
        Set SegmentRating = SegmentRating * ProductFactor Value
    Return SegmentRating if RatingType = "Segment"
    Set Rating = Rating + SegmentRating
Return Rating

2.4.2.2 Calculating the ProductFactor Value

If any RatingTableKey.FactorType for the RatingTable = E(mployee) and the Rating by Employee option was specified, then this procedure will be invoked for each employee. If it is not necessary to provide ratings by employee, then the looping through each employee will be done within this procedure. In particular, if a policy invokes bracketing for a group census factor, then the EGBRE cannot report ratings by employee.

Perform table lookup.
    For each RatingTableKey of the RatingTable of the ProductFactor (the number of RatingTableKeys is equal to RatingTable.Dimension).
    If any RatingTableKey.FactorType ="E" perform Employee loop as described above.
    If RatingTableKey.KeyType = "E"quality then find the table column with value equal to one of the following:
    If RatingTableKey.KeyType = "R" ange then find the table column with the smallest value which equals or exceeds one of the following:
    If RatingTableKey.KeyType = "L"ocation then find the table column with value which contains one of the following (not yet well defined):
    If RatingTableKey.KeyType = "T"rend then lookup the single table value without column key. Then calculate the number of months by subtracting the Product.TrendDate from the RatingDate. Exponentiate the table return value with the number of months.

        If RatingTableKey.FactorType = "E"mployee then use the EmployeeFactor.FactorValue where EmployeeFactor.FactorID = RatingTableKey.TableKey

        If RatingTableKey.FactorType = "C" then use the ConsumerFactor.FactorValue where ConsumerFactor.FactorID = RatingTableKey.TableKey

        If RatingTableKey.FactorType = "P" then one of several possible values can be used for PolicyOption.OptionValue. If there is a ConsumerOption.OptionValue for the coverage then set PolicyOption.OptionValue = ConsumerOption.OptionValue. If not then use the policy's PolicyOption.OptionValue. If there is none, set PolicyOption.OptionValue = Option.OptionValue where

DefaultYN = "Y". Then use the PolicyOption.OptionValue where PolicyOption.OptionID = RatingTableKey.TableKey

If RatingTableKey.FactorType = "R" then use the value stored from the previous table lookup.

With all column keys, perform the table lookup. Use the table DimX where X = RatingTable.Dimension
If Rating.NextRatingTable IS NULL then return DimX.RatingValue
Else
Perform Table Lookup, but use the table named in RatingTable.NextRatingTable
If RatingTable.NextRelationship = "M"ultiply then multiply the returned values from the two tables.
If RatingTable.NextRelationship = "E"xponentiate then take the first table's value to the second table's power.
If RatingTable.NextRelationship = "K"ey then use the first table's value as a key in the second table's lookup.

2.5 DATA ENTITIES

2.5.1 ENTITY RELATIONSHIP DIAGRAMS

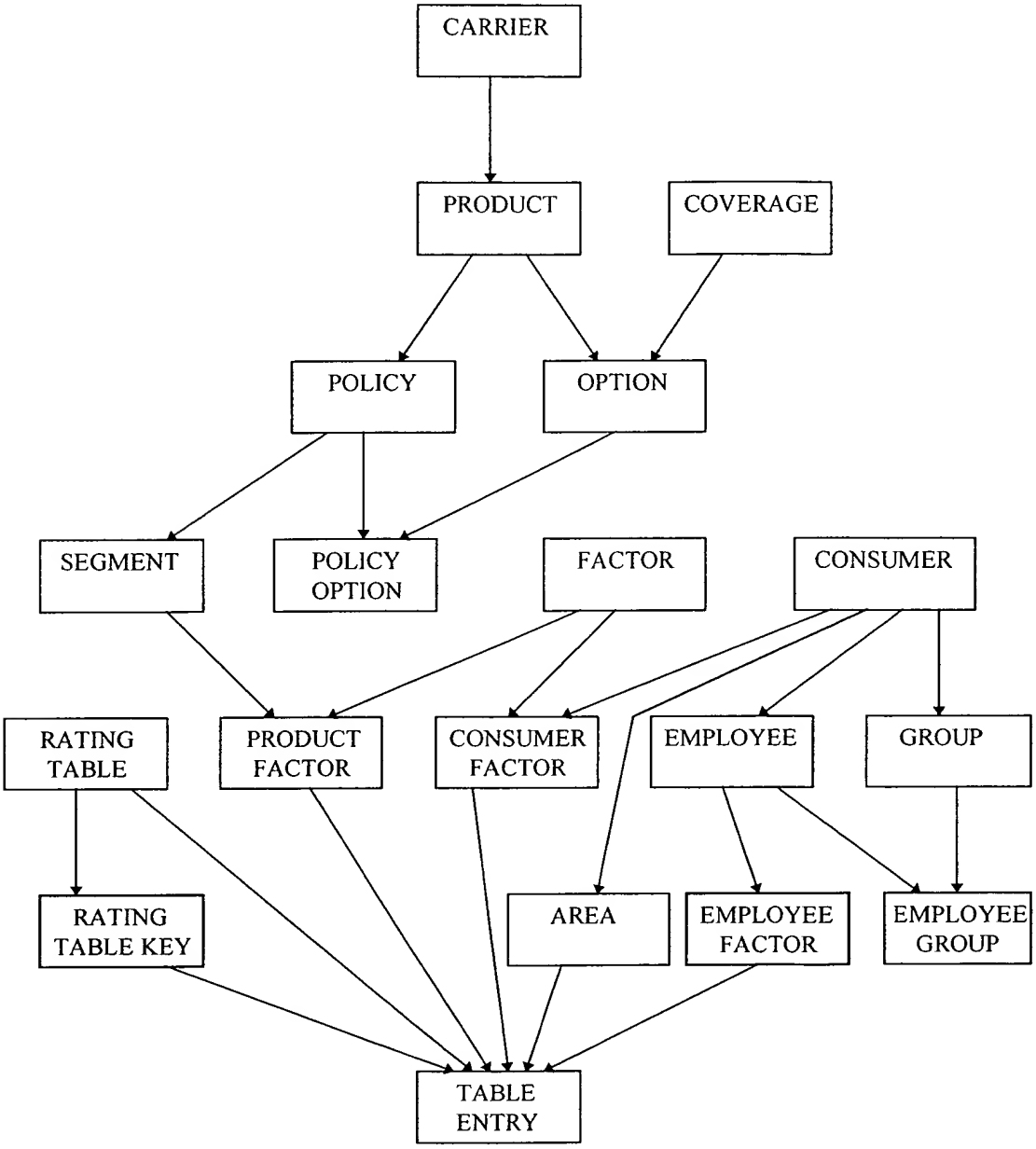

fig. 2 Data Entity Relationship Diagram

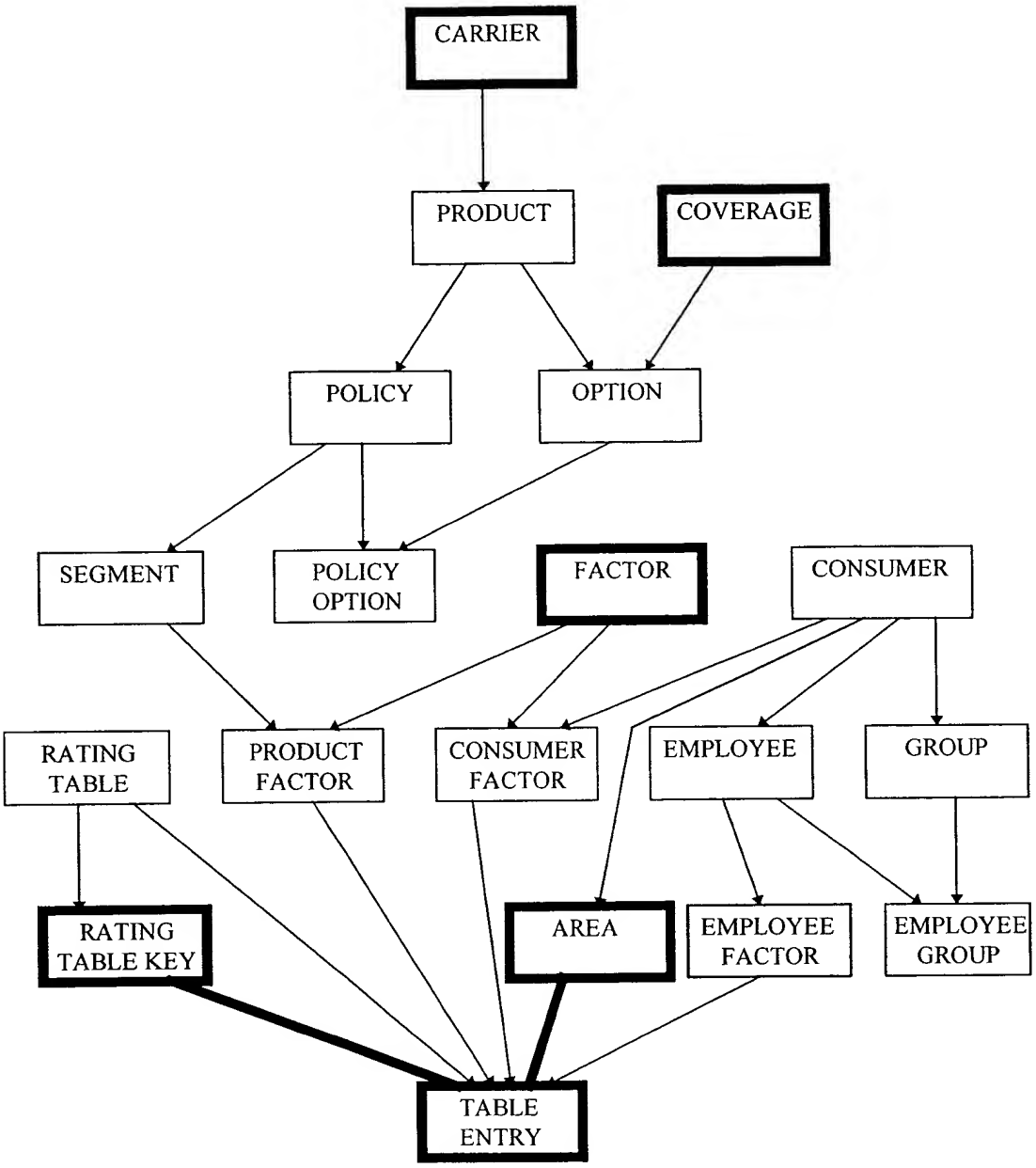

fig. 3 Administrative Setup Entity Relationship Diagram

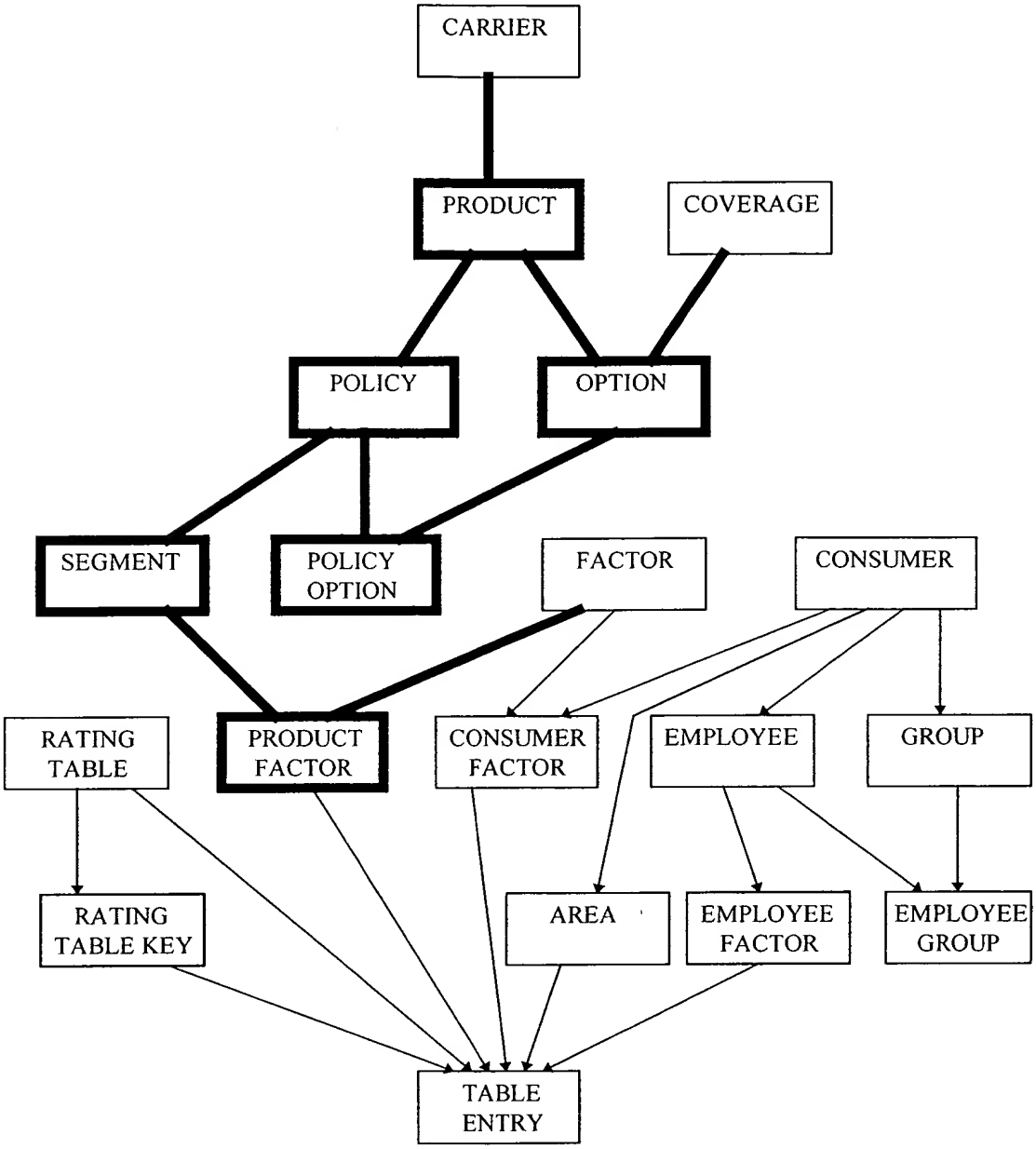

fig. 4 Plan Setup Entity Relationship Diagram

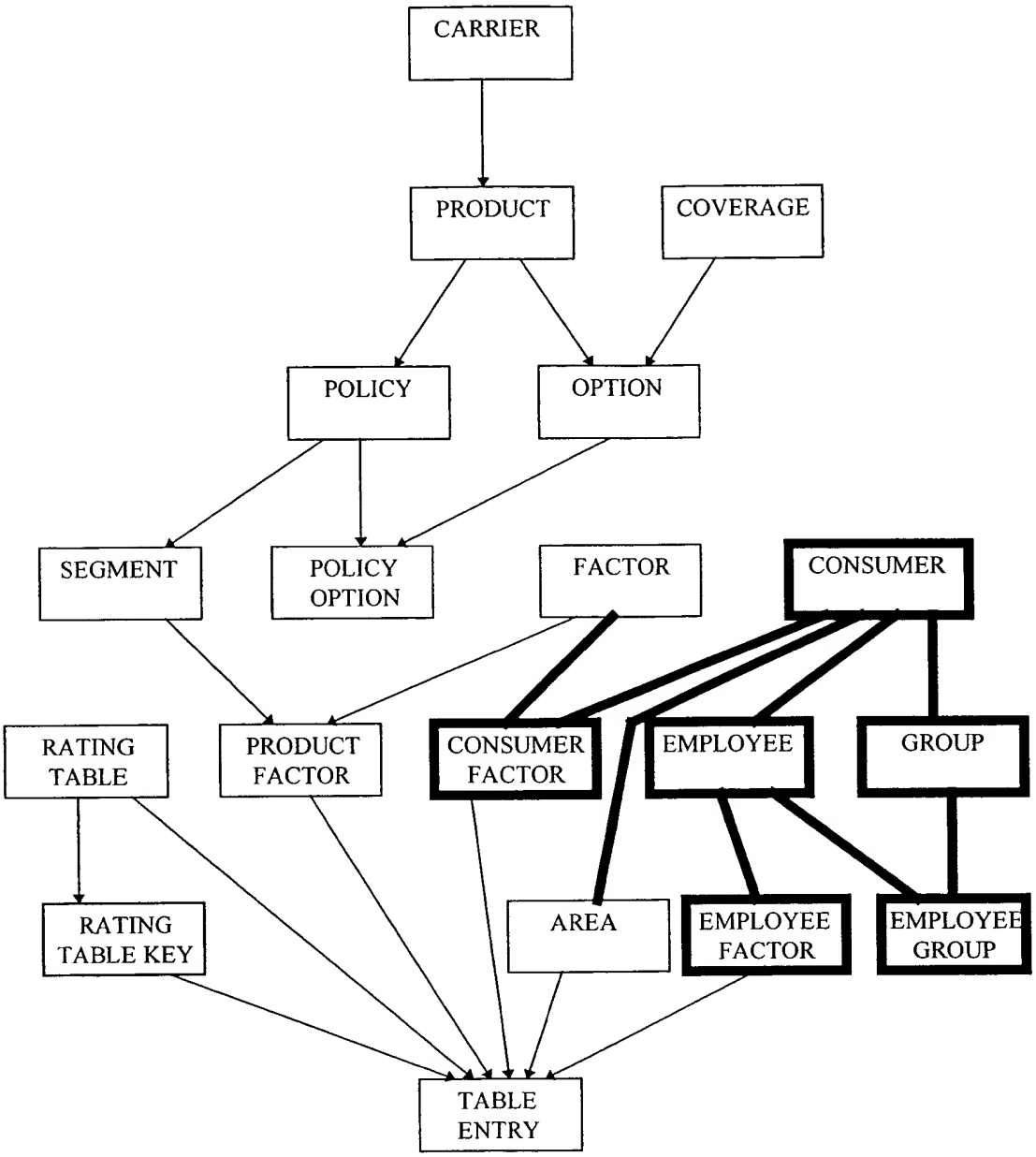

fig. 5 Consumer Setup Entity Relationship Diagram

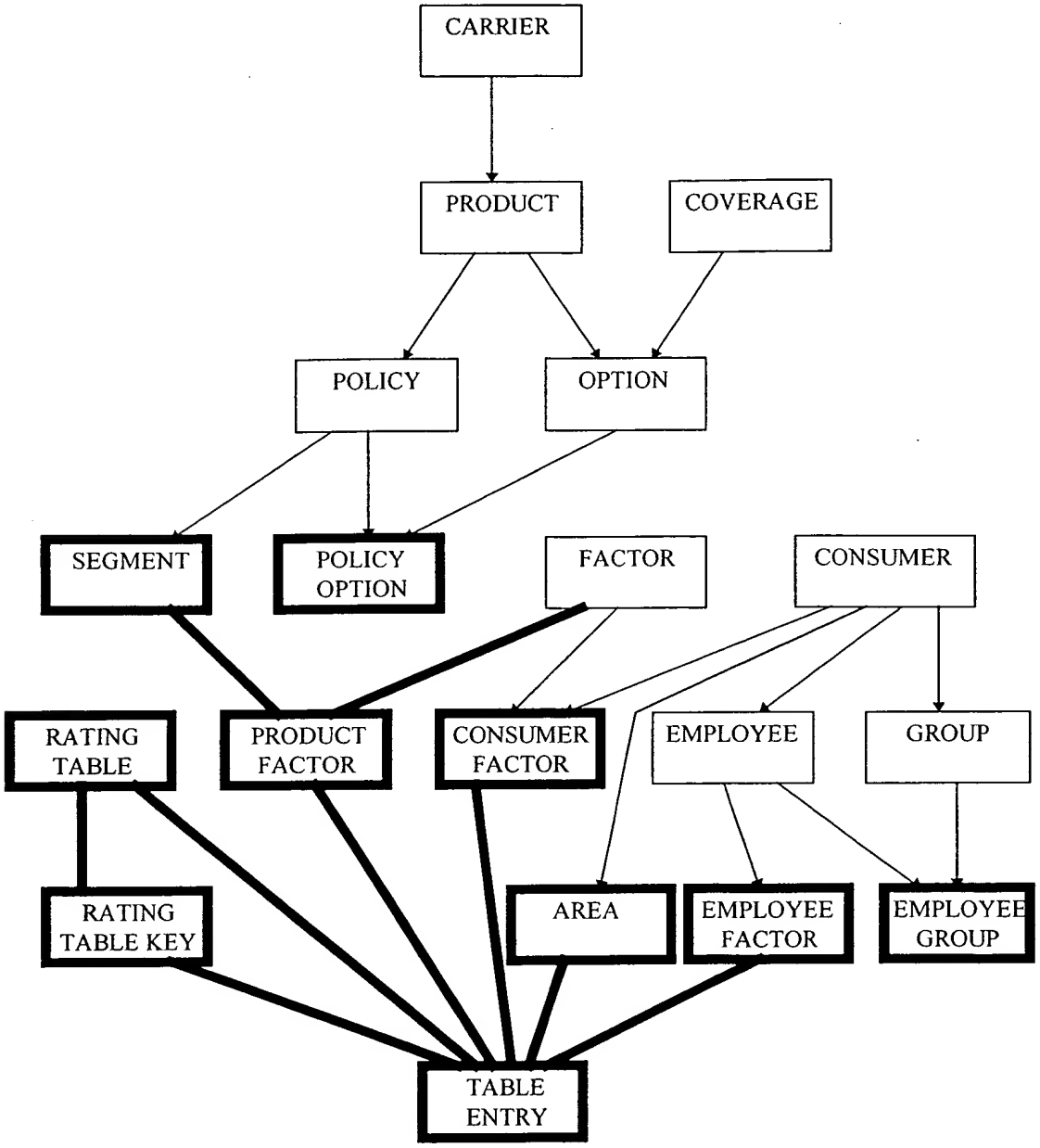

fig. 6 Rating Entity Relationship Diagram

2.5.2 DATA DICTIONARY

Please note that the data dictionary includes both requirements and design information. They are included together for convenience.

2.5.2.1 Carrier

Common Name: Carrier
Type: Object Class, Relation
Definition: Providers of the insurance products to whom the consumer is referred.
Synonyms: Insurance company, provider
Key Attributes: Carrier ID
Dependent Attributes: Carrier Name
Expected Key Values:

2.5.2.2 Product

Common Name: Product
Type: Object Class, Relation
Definition: Groups of coverages and a formula for evaluating them, bundled as a product of a carrier.
Synonyms: Plan
Key Attributes: Product ID
Dependent Attributes: Product Description
Expected Key Values:

2.5.2.3 Product Type

Common Name: Product Type
Type: Attribute of Product
Definition: Groups of insurance products according to regulations, type of network, and administrative style.
Synonyms:
Key Attributes: Product Type ID
Dependent Attributes: Product Type Description
Expected Key Values: HMO with gatekeeper PCP
HMO without gatekeeper PCP
POS in network requires gatekeeper
POS in network does not require gatekeeper
POS triple option
PPO
EPO
Indemnity with utilization review
Indemnity without utilization review

2.5.2.4 Insurance Type

Common Name: Insurance Type
Type: Attribute of Product
Definition: Type of insurance offered.
Synonyms:
Key Attributes:
Dependent Attributes:
Expected Key Values: Group medical
Short-term disability
Long-term disability
Long-term care

Group dental
Vision
Group life/accident/disability

2.5.2.5 Coverage

Common Name: Coverage
Type: Object Class, Relation
Definition: Attributes of a product affecting its value to the consumer. The consumer may select a value for each coverage (or they may be pre-selected by the carrier).
Synonyms: Coverage
Key Attributes: Coverage ID
Dependent Attributes: Coverage Description
Expected Key Values: Yearly deductible, individual
Yearly deductible, family
Yearly coinsurance
Yearly out of pocket maximum
Physician PCP services copay
Specialist services copay
Chiropractic services copay
Inpatient hospital copay
Outpatient hospital emergency room copay
Outpatient hospital surgery copay
Prescription drugs generic copay
Prescription drugs brand copay
Maternity
Mental health
Substance abuse
Durable medical equipment
First dollar accident coverage
Transplant coverage
Infertility drugs
Home health
Skilled nursing facility
Preventive coverages
Hospice coverages
Network
Group size
Geographic area

2.5.2.6 Option

Common Name: Option
Type: Object Class, Relation
Definition: Sets of two or more values of a coverage from which the consumer may make a selection. Different options affect the value and cost of a coverage.
Synonyms:
Key Attributes: Option ID
Dependent Attributes: Option Description
Expected Key Values: Various dollar amounts for deductibles and copayments. Yes or no for some other coverages.

2.5.2.7 Policy

Common Name: Policy
Type: Object Class, Relation
Definition: Specified set of options selected by the carrier and consumer.
Synonyms: Policies
Key Attributes: Policy ID
Dependent Attributes: Group minimum size
Group maximum size
Expected Key Values:

2.5.2.8 Network

Common Name: Network
Type: Attribute of Policy
Definition: Groups of service providers including hospitals and doctors.
Synonyms:
Key Attributes: Network ID
Dependent Attributes: Network Description
Expected Key Values:

2.5.2.9 Policy Option

Common Name: Policy Option
Type: Object Class, Relation
Definition: The specific options, one for each coverage, selected to be a product by a carrier and chosen by a consumer to comprise the specific product for which a rating may be made.
Synonyms:
Key Attributes: Policy ID, Option ID
Dependent Attributes:
Expected Key Values:

2.5.2.10 Segment

Common Name: Segment
Type: Object Class, Relation
Definition: A collection of coverages to which the same factors are applied. Each segment has a base factor which is to be multiplied by specified product and consumer factors to yield a segment rate. The sum of all segment rates is the consumer rating.
Synonyms:
Key Attributes: Policy ID, Sequence
Dependent Attributes: Rating Table ID
Expected Key Values:

2.5.2.11 Factor

Common Name: Factor
Type: Object Class, Relation
Definition: Attributes of consumers (or policies) which affect the rate which must be paid to purchase a fixed insurance product.
Synonyms:
Key Attributes: Policy ID, Factor ID
Dependent Attributes:
Expected Key Values: Base Factor (30 year old male normalization)
Expenses
Blend

2.5.2.12 Consumer

Common Name: Consumer
Type: Object Class, Relation
Definition: An expected purchaser of insurance who uses the InsWeb software to obtain comparative insurance rates.
Synonyms: Customer, insured
Key Attributes: Consumer ID
Dependent Attributes:
Expected Key Values:

2.5.2.13 Consumer Factor

Common Name: Consumer Factor
Type: Object Class, Relation
Definition: The values for relevant factors which are true for a particular consumer.
Synonyms: Factor
Key Attributes: Consumer ID, Factor ID
Dependent Attributes: Factor Value
Expected Key Values: Geographic
Inflationary trend
Group size
Industry
Takeover / pre-existing carrier

2.5.2.14 Employee

Common Name: Employee
Type: Object Class, Relation
Definition: One who works for a consumer and participates in one or more insurance products.
Synonyms: member
Key Attributes: Employee ID (which may be real, but may be an internally used pseudonym).
Dependent Attributes:
Expected Key Values:

2.5.2.15 Employee Factor

Common Name: Employee Factor
Type: Object Class, Relation
Definition: The values for relevant factors which are true for a particular employee.
Synonyms: Factor
Key Attributes: Consumer ID, Factor ID
Dependent Attributes: Factor Value
Expected Key Values: Age / sex / family
Healthy lifestyle / smoker (may depend on age / sex)

2.5.2.16 Product Factor

Common Name: Product Factor
Type: Object Class, Relation
Definition: The set of product factors which apply to each segment.
Synonyms:
Key Attributes: Policy ID, Sequence, Factor ID
Dependent Attributes: Table ID
Expected Key Values:

2.5.2.17 Rating Table

Common Name: Rating Table
Type: Object Class, Relation
Definition: A collection of numeric entries accessed individually by the Table ID and one or more Factor Ids.
Synonyms: Table
Key Attributes: Table ID
Dependent Attributes:
Expected Key Values:

2.5.2.18 Rating Table Entry

Common Name: Rating Table Entry
Type: Object Class, Relation
Definition: A number. Accessed with the name of the table which contains it and one or more customer or product factors which are used to define where in the table the number resides.
Synonyms:
Key Attributes:
Dependent Attributes:
Expected Key Values:

2.6 PERFORMANCE

The following assumptions are made. These parameters may change in the course of the design process of the balance of the Employee Group Benefits application.

2.6.1 RATING SPEED

The rating engine should be able to perform a single request for 100 ratings within three seconds on an otherwise idle, well configured, single Pentium 2 - 200 megahertz CPU server, while in mid-stream operation (i.e. not having just started). It is expected that as usage increases and these numbers are exceeded, it will be cost effective to employ more elaborate hardware.

2.6.2 TRANSACTION CAPACITY

The EGBRE must be able to provide ratings for up to twenty separate requests per minute, each with up to 100 policy ratings to be performed.

2.6.3 ACCURACY

The rating engine will handle rate adjustment factors from 0.000001 to 999999.999999. Dollar values of up to $999,999.00 for single table entries will be allowed and total ratings of up to $99,999,999.00 will be handled. All calculations will be done to the accuracy of the table entries as entered without intermediate rounding or truncating to less than nine significant digits, and the final rating will then be rounded (not truncated) to the nearest penny for each employee and segment. Calculated ratings will not vary from the carrier's calculated rating by more than one cent per employee and segment (barring carrier error).

2.6.4 RELIABILITY

Where products are configured within the EGBRE without modification and all rating tables are correctly entered, the EGBRE rating must be the same as the correct rating to within one cent every time. Where it is necessary to modify the carrier's product to meet InsWeb's requirements the returned rating must be within one percent of the manual rating ninety-nine percent of the time. Carrier's will be expected to submit test policies and risk profiles for verification testing.

2.6.5 MAINTAINABILITY

The EGBRE must be developed using standard programming languages and architecture adopted by InsWeb, including but not limited to Visual Basic 5.0, C++ 4.0, SQL Server 6.5, OLE (DCOM), and Clink/Slink.

2.6.6 DATA VOLUME

The system should be able to support the following data volumes without significant performance degradation (25% reduction of ideal condition). The numbers are meant to represent typical values for performance calculations and are not to be used as design or implementation limits. By design, there will be no limit on the numbers of records except where indicated below other than practical considerations such as total disk space available, performance degradation, and the user's ability to enter unique names long enough to permit large numbers of values.

2.6.6.1 Carriers

20 Carriers

2.6.6.2 Products

5 Products per Carrier = 100 Products

2.6.6.3 Coverage

100 Coverage records

2.6.6.4 Policies

100 Policies per Product = 10,000 Policies (32,767 Policies per Product maximum)

2.6.6.5 Options

20 Coverages per Product and 4 Options per Coverage = 8,000 Options (32,767 Options per Coverage maximum)

2.6.6.6 Segments

2 Segments per Policy = 20,000 Segments (2,147,483,647 Segments maximum)

2.6.6.7 Policy Options

1 Policy Option per Coverage and 20 Coverages per Policy = 200,000 Policy Options

2.6.6.8 Factors

50 Factors

2.6.6.9 Consumers

10,000 Consumers per year. Consumer data kept online for one year.

2.6.6.10 Rating Tables

100 Tables per Carrier = 10,000 Tables

2.6.6.11 Product Factors

5 Factors per Segment = 100,000 Segment Factors

2.6.6.12 Consumer Factors

20 Factors per Consumer (Employer) = 200,000 Consumer Factors

2.6.6.13 Employees

20 Employees per Consumer = 200,000 Employees

2.6.6.14 Employee Factors

10 Employee Factors per Employee = 2,000,000 Employee Factors

2.6.6.15 Groups

4 Groups per Consumer = 40,000 Groups

2.6.6.16 Group Employees

20 Employees per Group (or 4 Groups per Employee)= 800,000 Group Employee records

2.6.6.17 Rating Table Entries

100 Entries per Table = 1,000,000 Rating Table Entries

2.6.7 TOTAL DATA STORAGE

With less than 5,000,000 records and less than 100 bytes per record for the high population tables, and multiplying by two to account for indices and other DBMS overhead, data storage use will be less than 1,000 megabytes. A soon to be released version of Microsoft SQL Server will be able to manage up to 3 gigabytes of data in memory, so it is likely that this application will be able to scale substantially beyond these initial projections with excellent performance.

3. DESIGN SPECIFICATION

3.1 OBJECT MODEL

The Employee Group Benefits Rating Engine will be implemented as a software object which is capable of communicating with other InsWeb objects. Objects of concern to the EGBRE fall into two categories:

- Calling Objects - objects which request data from the EGBRE.
- Passed Objects - objects passed to the EGBRE which are responsible for data necessary to the rating calculation.

3.1.1 CALLING OBJECTS

The EGBRE interface method for calling objects is:

Rating = EGBRE.GetRating(RiskProfile, Policy, RatingDateTime, RatingType)

where the RiskProfile object includes all data for the consumer and the consumer's employees, and the Policy object includes high level policy information. The RatingDateTime is the time the rating is to be effective. Tables used by the EGBRE will be the ones in effect at that time. The RatingType specifies whether employee and segment ratings are to be returned.

Upon completion of the calculation, the rating information is returned to the Rating variable of the calling object.

3.1.2 PASSED OBJECTS

The RiskProfile and Policy objects are passed to the EGBRE with each request for a rating.

The EGBRE will retrieve policy information as required from the Policy object via the methods which it makes available. As each policy requires, the EGBRE will request consumer and employee information from the RequestProfile object via its methods. The Interfaces section above includes lists of risk profile data which is likely to be used, including consumer factors, employee factors, and policy options which will be part of the rating calculation.

The details of the actual calculation are outlined in the algorithms section. The general approach is that the EGBRE determines structural details of the policy, then uses the necessary consumer information to look up rating factors in the tables specified by the product. The EGBRE uses information from the policy to determine how the calculation proceeds with the numbers returned from the table lookups.

3.2 DATABASE

3.2.1 SCHEMA

The table structure is intended to serve the needs of all components of the Employee Group Benefits application. However, the fields and data types specified here are only those necessary to serve the rating engine. As the needs of other components become better defined appropriate fields will be modified or added.

3.2.1.1 Database Preparation

```
IF EXISTS (SELECT * FROM sysobjects WHERE id = object_id('EBRatingHistoryDetail'))
    DROP TABLE EBRatingHistoryDetail
IF EXISTS (SELECT * FROM sysobjects WHERE id = object_id('EBRatingHistory'))
    DROP TABLE EBRatingHistory
IF EXISTS (SELECT * FROM sysobjects WHERE id = object_id('EBArea '))
    DROP TABLE EBArea
IF EXISTS (SELECT * FROM sysobjects WHERE id = object_id('EBDim5'))
    DROP TABLE EBDim5
IF EXISTS (SELECT * FROM sysobjects WHERE id = object_id('EBDim4'))
    DROP TABLE EBDim4
IF EXISTS (SELECT * FROM sysobjects WHERE id = object_id('EBDim3'))
    DROP TABLE EBDim3
IF EXISTS (SELECT * FROM sysobjects WHERE id = object_id('EBDim2'))
    DROP TABLE EBDim2
IF EXISTS (SELECT * FROM sysobjects WHERE id = object_id('EBDim1'))
    DROP TABLE EBDim1
IF EXISTS (SELECT * FROM sysobjects WHERE id = object_id('EBDim0'))
    DROP TABLE EBDim0
IF EXISTS (SELECT * FROM sysobjects WHERE id = object_id('EBRatingTableKey'))
    DROP TABLE EBRatingTableKey
IF EXISTS (SELECT * FROM sysobjects WHERE id = object_id('EBRatingTable'))
    DROP TABLE EBRatingTable
IF EXISTS (SELECT * FROM sysobjects WHERE id = object_id(' EBProductFactor '))
    DROP TABLE EBProductFactor
IF EXISTS (SELECT * FROM sysobjects WHERE id = object_id('EBEmployeeGroup'))
    DROP TABLE EBEmployeeGroup
IF EXISTS (SELECT * FROM sysobjects WHERE id = object_id('EBGroup'))
    DROP TABLE EBGroup
IF EXISTS (SELECT * FROM sysobjects WHERE id = object_id('EBEmployeeFactor'))
    DROP TABLE EBEmployeeFactor
IF EXISTS (SELECT * FROM sysobjects WHERE id = object_id('EBEmployee'))
    DROP TABLE EBEmployee
IF EXISTS (SELECT * FROM sysobjects WHERE id = object_id('EBConsumerFactor'))
    DROP TABLE EBConsumerFactor
IF EXISTS (SELECT * FROM sysobjects WHERE id = object_id('EBConsumer'))
    DROP TABLE EBConsumer
IF EXISTS (SELECT * FROM sysobjects WHERE id = object_id('EBFactor'))
    DROP TABLE EBFactor
IF EXISTS (SELECT * FROM sysobjects WHERE id = object_id('EBSegment'))
    DROP TABLE EBSegment
IF EXISTS (SELECT * FROM sysobjects WHERE id = object_id('EBPolicyOption'))
    DROP TABLE EBPolicyOption
IF EXISTS (SELECT * FROM sysobjects WHERE id = object_id('EBPolicy'))
```

```
    DROP TABLE EBPolicy
IF EXISTS (SELECT * FROM sysobjects WHERE id = object_id('EBOption'))
    DROP TABLE EBOption
IF EXISTS (SELECT * FROM sysobjects WHERE id = object_id('EBCoverage'))
    DROP TABLE EBCoverage
IF EXISTS (SELECT * FROM sysobjects WHERE id = object_id('EBProduct'))
    DROP TABLE EBProduct
IF EXISTS (SELECT * FROM sysobjects WHERE id = object_id(' EBInsuranceType '))
    DROP TABLE EBInsuranceType
IF EXISTS (SELECT * FROM sysobjects WHERE id = object_id('EBCarrier'))
    DROP TABLE EBCarrier
GO
```

3.2.1.2 Table Definitions

3.2.1.2.1 CREATE TABLE EBCarrier

```
(   CarrierID           VARCHAR(16)     NOT NULL,

    CONSTRAINT PKCarrier
        PRIMARY KEY CLUSTERED (CarrierID)
)
```

3.2.1.2.2 CREATE TABLE EBInsuranceType

```
(   InsuranceType       VARCHAR(16)     NOT NULL,

    CONSTRAINT PKInsuranceType
        PRIMARY KEY CLUSTERED (InsuranceType)
)
```

3.2.1.2.3 CREATE TABLE EBProduct

```
(   CarrierID           VARCHAR(16)     NOT NULL,
    ProductID           VARCHAR(16)     NOT NULL,
    InsuranceType       VARCHAR(16)     NOT NULL,
    TrendDate           DATETIME        NULL,

    CONSTRAINT PKProduct
        PRIMARY KEY CLUSTERED (CarrierID, ProductID),

    CONSTRAINT FKCarrierofProduct
        FOREIGN KEY (CarrierID)
        REFERENCES EBCarrier(CarrierID),

    CONSTRAINT FKInsuranceTypeofProduct
        FOREIGN KEY (InsuranceType)
        REFERENCES EBInsuranceType(InsuranceType)
)
```

3.2.1.2.4 CREATE TABLE EBCoverage

```
(   CoverageID          VARCHAR(16)     NOT NULL,

    CONSTRAINT PKCoverage
        PRIMARY KEY CLUSTERED (CoverageID)
)
```

3.2.1.2.5 CREATE TABLE EBOption

```
(   CarrierID       VARCHAR(16)     NOT NULL,
    ProductID       VARCHAR(16)     NOT NULL,
    CoverageID      VARCHAR(16)     NOT NULL,
    OptionNo        SMALLINT        NOT NULL,
    OptionValue     VARCHAR(64)     NULL,
    Units           VARCHAR(16)     NULL,
    DefaultYN       CHAR(1)         NOT NULL    DEFAULT "N",

    CONSTRAINT PKOption
        PRIMARY KEY CLUSTERED (CarrierID, ProductID, CoverageID, OptionNo),

    CONSTRAINT FKProductofOption
        FOREIGN KEY (CarrierID, ProductID)
        REFERENCES EBProduct(CarrierID, ProductID),

    CONSTRAINT FKCoverageofOption
        FOREIGN KEY (CoverageID)
        REFERENCES EBCoverage(CoverageID),

    Constraint CKDefaultYNofOption
        CHECK (DefaultYN IN ("Y", "N"))
)
```

3.2.1.2.6 CREATE TABLE EBPolicy

```
(   CarrierID       VARCHAR(16)     NOT NULL,
    ProductID       VARCHAR(16)     NOT NULL,
    PolicyNo        SMALLINT        NOT NULL,

    CONSTRAINT PKPolicy
        PRIMARY KEY CLUSTERED (CarrierID, ProductID, PolicyNo),

    CONSTRAINT FKProductofPolicy
        FOREIGN KEY (CarrierID, ProductID)
        REFERENCES EBProduct(CarrierID, ProductID)
)
```

3.2.1.2.7 CREATE TABLE EBPolicyOption

```
(   CarrierID       VARCHAR(16)     NOT NULL,
    ProductID       VARCHAR(16)     NOT NULL,
    PolicyNo        SMALLINT        NOT NULL,
    CoverageID      VARCHAR(16)     NOT NULL,
    OptionNo        SMALLINT        NOT NULL,
    OptionValue     VARCHAR(64)     NOT NULL,

    CONSTRAINT PKPolicyOption
        PRIMARY KEY CLUSTERED (CarrierID, ProductID, PolicyNo, CoverageID, OptionNo),

    CONSTRAINT FKPolicyofPolicyOption
        FOREIGN KEY (CarrierID, ProductID, PolicyNo)
        REFERENCES EBPolicy(CarrierID, ProductID, PolicyNo),

    CONSTRAINT FKOptionofPolicyOption
        FOREIGN KEY (CarrierID, ProductID, CoverageID, OptionNo)
        REFERENCES EBOption(CarrierID, ProductID, CoverageID, OptionNo)
)
```

3.2.1.2.8 CREATE TABLE EBSegment

```
(   CarrierID       VARCHAR(16)     NOT NULL,
    ProductID       VARCHAR(16)     NOT NULL,
    PolicyNo        SMALLINT        NOT NULL,
    SegmentNo       INTEGER         NOT NULL    IDENTITY,
    BaseValue       NUMERIC(12, 6)  NOT NULL    DEFAULT 1,
    SeparateYN      CHAR(1)         NOT NULL    DEFAULT 'N',

    CONSTRAINT PKSegment
        PRIMARY KEY CLUSTERED (CarrierID, ProductID, PolicyNo, SegmentNo),

    CONSTRAINT UNSegmentNo
        UNIQUE NONCLUSTERED (SegmentNo),

    CONSTRAINT FKPolicyofSegment
        FOREIGN KEY (CarrierID, ProductID, PolicyNo)
        REFERENCES EBPolicy(CarrierID, ProductID, PolicyNo),

    Constraint CKSeparateYNofSegment
        CHECK (SeparateYN IN ("Y", "N"))
)
```

3.2.1.2.9 CREATE TABLE EBFactor

```
(   FactorID        VARCHAR(16)     NOT NULL,
    Units           VARCHAR(16)     NULL,

    CONSTRAINT PKFactor
        PRIMARY KEY CLUSTERED (FactorID)
)
```

3.2.1.2.10 CREATE TABLE EBConsumer

```
(   ConsumerID          VARCHAR(16)     NOT NULL,
    ConsumerAlias       VARCHAR(16)     NULL,

    CONSTRAINT PKConsumer
        PRIMARY KEY CLUSTERED (ConsumerID)
)
```

3.2.1.2.11 CREATE TABLE EBConsumerFactor

```
(   ConsumerID          VARCHAR(16)     NOT NULL,
    FactorID            VARCHAR(16)     NOT NULL,
    FactorValue         VARCHAR(255)    NULL,

    CONSTRAINT PKConsumerFactor
        PRIMARY KEY CLUSTERED (ConsumerID, FactorID),

    CONSTRAINT FKFactorofConsumerFactor
        FOREIGN KEY (FactorID)
        REFERENCES EBFactor(FactorID),

    CONSTRAINT FKConsumerofConsumerFactor
        FOREIGN KEY (ConsumerID)
        REFERENCES EBConsumer(ConsumerID)
)
```

3.2.1.2.12 CREATE TABLE EBEmployee

```
(   ConsumerID          VARCHAR(16)     NOT NULL,
    EmployeeID          VARCHAR(16)     NOT NULL,

    CONSTRAINT PKEmployee
        PRIMARY KEY CLUSTERED (ConsumerID, EmployeeID)
)
```

3.2.1.2.13 CREATE TABLE EBEmployeeFactor

```
(   ConsumerID          VARCHAR(16)     NOT NULL,
    EmployeeID          VARCHAR(16)     NOT NULL,
    FactorID            VARCHAR(16)     NOT NULL,
    FactorValue         VARCHAR(255)    NULL,

    CONSTRAINT PKEmployeeFactor
        PRIMARY KEY CLUSTERED (ConsumerID, EmployeeID, FactorID),

    CONSTRAINT FKFactorofEmployeeFactor
        FOREIGN KEY (FactorID)
        REFERENCES EBFactor(FactorID),

    CONSTRAINT FKEmployeeofEmployeeFactor
        FOREIGN KEY (ConsumerID, EmployeeID)
        REFERENCES EBEmployee(ConsumerID, EmployeeID)
)
```

3.2.1.2.14 CREATE TABLE EBGroup

```
(   ConsumerID      VARCHAR(16)     NOT NULL,
    GroupID         VARCHAR(16)     NOT NULL,

    CONSTRAINT PKGroup
        PRIMARY KEY CLUSTERED (ConsumerID, GroupID)
)
```

3.2.1.2.15 CREATE TABLE EBEmployeeGroup

```
(   ConsumerID      VARCHAR(16)     NOT NULL,
    EmployeeID      VARCHAR(16)     NOT NULL,
    GroupID         VARCHAR(16)     NOT NULL,

    CONSTRAINT PKEmployeeGroup
        PRIMARY KEY CLUSTERED (ConsumerID, EmployeeID, GroupID),

    CONSTRAINT FKGroupofEmployeeGroup
        FOREIGN KEY (ConsumerID, GroupID)
        REFERENCES EBGroup(ConsumerID, GroupID),

    CONSTRAINT FKEmployeeofEmployeeGroup
        FOREIGN KEY (ConsumerID, EmployeeID)
        REFERENCES EBEmployee(ConsumerID, EmployeeID)
)
```

3.2.1.2.16 CREATE TABLE EBProductFactor

```
(   SegmentNo       INTEGER         NOT NULL,
    FactorID        VARCHAR(16)     NOT NULL,
    RatingTableID   VARCHAR(16)     NULL,
    BaseValue       NUMERIC(12, 6)  NOT NULL    DEFAULT 1,
    Algorithm       CHAR(1)         NOT NULL    DEFAULT "N",
    MinimumValue    NUMERIC(12, 6)  NOT NULL    DEFAULT 1,
    MaximumValue    NUMERIC(12, 6)  NOT NULL    DEFAULT 1,

    CONSTRAINT PKProductFactor
        PRIMARY KEY CLUSTERED (SegmentNo, FactorID),

    CONSTRAINT FKFactorofProductFactor
        FOREIGN KEY (FactorID)
        REFERENCES EBFactor(FactorID),

    CONSTRAINT FKSegmentofProductFactor
        FOREIGN KEY (SegmentNo)
        REFERENCES EBSegment(SegmentNo),

    Constraint CKAlgorithmofProductFactor
        CHECK (Algorithm IN ("T", "N"))
            --T = Table
            --N = None
)
```

3.2.1.2.17 CREATE TABLE EBRatingTable

```
(   RatingTableID       VARCHAR(16)     NOT NULL,
    RatingTableVersion  SMALLINT        NOT NULL    DEFAULT 1,
    ExpirationDate      DATETIME        NOT NULL,
    Dimension           TINYINT         NOT NULL    DEFAULT 1,
    NextRatingTable VARCHAR(16)     NULL,
    NextRelationship    CHAR(1)         NOT NULL    DEFAULT "N",

    CONSTRAINT PKRatingTable
        PRIMARY KEY CLUSTERED (RatingTableID, RatingTableVersion),

    Constraint CKNextRelationofRatingTable
        CHECK (NextRelationship IN ("N", "M", "E", "K"))
            -- N = None
            -- M = Multiply
            -- E = Exponentiate
            -- K = Key
)
```

3.2.1.2.18 CREATE TABLE EBRatingTableKey

```
(   RatingTableID       VARCHAR(16)     NOT NULL,
    TableKey            VARCHAR(16)     NOT NULL,
    FactorType          CHAR(1)         NOT NULL    DEFAULT "C",
    KeyType             CHAR(1)         NOT NULL    DEFAULT "E",

    CONSTRAINT PKRatingTableKey
        PRIMARY KEY CLUSTERED (RatingTableID, TableKey),

    Constraint CKFactorTypeofRatingTableKey
        CHECK (FactorType IN ("P", "C", "E", "R")),
            -- P = Product
            -- C = Consumer
            -- E = Employee
            -- R = Rating Table

    Constraint CKKeyTypeofRatingTableKey
        CHECK (KeyType IN ("E", "R", "L", "T"))
            -- E = Equality
            -- R = Range
            -- L = Location
            -- T = Trend
)
```

3.2.1.2.19 CREATE TABLE EBDim0

```
(   RatingTableID       VARCHAR(16)     NOT NULL,
    RatingTableVersion  SMALLINT        NOT NULL,
    RatingValue         NUMERIC(12, 6)  NOT NULL    DEFAULT 1,

    CONSTRAINT PKDim0
        PRIMARY KEY CLUSTERED (RatingTableID, RatingTableVersion),

    CONSTRAINT FKRatingTableofDim0
        FOREIGN KEY (RatingTableID, RatingTableVersion)
        REFERENCES EBRatingTable(RatingTableID, RatingTableVersion)
)
```

3.2.1.2.20 CREATE TABLE EBDim1

```
(   RatingTableID       VARCHAR(16)     NOT NULL,
    RatingTableVersion  SMALLINT        NOT NULL,
    TableKey1           VARCHAR(16)     NOT NULL,
    RatingValue         NUMERIC(12, 6)  NOT NULL    DEFAULT 1,

    CONSTRAINT PKDim1
        PRIMARY KEY CLUSTERED (RatingTableID, RatingTableVersion, TableKey1),

    CONSTRAINT FKRatingTableofDim1
        FOREIGN KEY (RatingTableID, RatingTableVersion)
        REFERENCES EBRatingTable(RatingTableID, RatingTableVersion),

    CONSTRAINT FKRatingTableKey1ofDim1
        FOREIGN KEY (RatingTableID, TableKey1)
        REFERENCES EBRatingTableKey(RatingTableID, TableKey)
)
```

3.2.1.2.21 CREATE TABLE EBDim2

```
(   RatingTableID       VARCHAR(16)     NOT NULL,
    RatingTableVersion  SMALLINT        NOT NULL,
    TableKey1           VARCHAR(16)     NOT NULL,
    TableKey2           VARCHAR(16)     NOT NULL,
    RatingValue         NUMERIC(12, 6)  NOT NULL    DEFAULT 1,

    CONSTRAINT PKDim2
        PRIMARY KEY CLUSTERED (RatingTableID, RatingTableVersion, TableKey1, TableKey2),

    CONSTRAINT FKRatingTableofDim2
        FOREIGN KEY (RatingTableID, RatingTableVersion)
        REFERENCES EBRatingTable(RatingTableID, RatingTableVersion),
    CONSTRAINT FKRatingTableKey1ofDim2T
        FOREIGN KEY (RatingTableID, TableKey1)
        REFERENCES EBRatingTableKey(RatingTableID, TableKey),
    CONSTRAINT FKRatingTableKey2ofDim2
        FOREIGN KEY (RatingTableID, TableKey2)
        REFERENCES EBRatingTableKey(RatingTableID, TableKey)
)
```

3.2.1.2.22 CREATE TABLE EBDim3

```
(   RatingTableID       VARCHAR(16)     NOT NULL,
    RatingTableVersion  SMALLINT        NOT NULL,
    TableKey1           VARCHAR(16)     NOT NULL,
    TableKey2           VARCHAR(16)     NOT NULL,
    TableKey3           VARCHAR(16)     NOT NULL,
    RatingValue         NUMERIC(12, 6)  NOT NULL    DEFAULT 1,

    CONSTRAINT PKDim3
        PRIMARY KEY CLUSTERED (RatingTableID, RatingTableVersion, TableKey1, TableKey2,
        TableKey3),

    CONSTRAINT FKRatingTableofDim3
        FOREIGN KEY (RatingTableID, RatingTableVersion)
        REFERENCES EBRatingTable(RatingTableID, RatingTableVersion),

    CONSTRAINT FKRatingTableKey1ofDim3
        FOREIGN KEY (RatingTableID, TableKey1)
        REFERENCES EBRatingTableKey(RatingTableID, TableKey),

    CONSTRAINT FKRatingTableKey2ofDim3
        FOREIGN KEY (RatingTableID, TableKey2)
        REFERENCES EBRatingTableKey(RatingTableID, TableKey),

    CONSTRAINT FKRatingTableKey3ofDim3
        FOREIGN KEY (RatingTableID, TableKey3)
        REFERENCES EBRatingTableKey(RatingTableID, TableKey)
)
```

3.2.1.2.23 CREATE TABLE EBDim4

```
(   RatingTableID       VARCHAR(16)     NOT NULL,
    RatingTableVersion  SMALLINT        NOT NULL,
    TableKey1           VARCHAR(16)     NOT NULL,
    TableKey2           VARCHAR(16)     NOT NULL,
    TableKey3           VARCHAR(16)     NOT NULL,
    TableKey4           VARCHAR(16)     NOT NULL,
    RatingValue         NUMERIC(12, 6)  NOT NULL    DEFAULT 1,

    CONSTRAINT PKDim4
        PRIMARY KEY CLUSTERED (RatingTableID, RatingTableVersion, TableKey1, TableKey2,
        TableKey3, TableKey4),

    CONSTRAINT FKRatingTableofDim4
        FOREIGN KEY (RatingTableID, RatingTableVersion)
        REFERENCES EBRatingTable(RatingTableID, RatingTableVersion),

    CONSTRAINT FKRatingTableKey1ofDim4
        FOREIGN KEY (RatingTableID, TableKey1)
        REFERENCES EBRatingTableKey(RatingTableID, TableKey),

    CONSTRAINT FKRatingTableKey2ofDim4
        FOREIGN KEY (RatingTableID, TableKey2)
        REFERENCES EBRatingTableKey(RatingTableID, TableKey),
```

```
    CONSTRAINT FKRatingTableKey3ofDim4
        FOREIGN KEY (RatingTableID, TableKey3)
        REFERENCES EBRatingTableKey(RatingTableID, TableKey),

    CONSTRAINT FKRatingTableKey4ofDim4
        FOREIGN KEY (RatingTableID, TableKey4)
        REFERENCES EBRatingTableKey(RatingTableID, TableKey)
)
```

3.2.1.2.24 CREATE TABLE EBDim5

```
(   RatingTableID       VARCHAR(16)     NOT NULL,
    RatingTableVersion  SMALLINT        NOT NULL,
    TableKey1           VARCHAR(16)     NOT NULL,
    TableKey2           VARCHAR(16)     NOT NULL,
    TableKey3           VARCHAR(16)     NOT NULL,
    TableKey4           VARCHAR(16)     NOT NULL,
    TableKey5           VARCHAR(16)     NOT NULL,
    RatingValue         NUMERIC(12, 6)  NOT NULL    DEFAULT 1,

    CONSTRAINT PKDim5
        PRIMARY KEY CLUSTERED (RatingTableID, RatingTableVersion, TableKey1, TableKey2,
        TableKey3, TableKey4, TableKey5),

    CONSTRAINT FKRatingTableofDim5
        FOREIGN KEY (RatingTableID, RatingTableVersion)
        REFERENCES EBRatingTable(RatingTableID, RatingTableVersion),

    CONSTRAINT FKRatingTableKey1ofDim5
        FOREIGN KEY (RatingTableID, TableKey1)
        REFERENCES EBRatingTableKey(RatingTableID, TableKey),

    CONSTRAINT FKRatingTableKey2ofDim5
        FOREIGN KEY (RatingTableID, TableKey2)
        REFERENCES EBRatingTableKey(RatingTableID, TableKey),

    CONSTRAINT FKRatingTableKey3ofDim5
        FOREIGN KEY (RatingTableID, TableKey3)
        REFERENCES EBRatingTableKey(RatingTableID, TableKey),

    CONSTRAINT FKRatingTableKey4ofDim5
        FOREIGN KEY (RatingTableID, TableKey4)
        REFERENCES EBRatingTableKey(RatingTableID, TableKey),

    CONSTRAINT FKRatingTableKey5ofDim5
        FOREIGN KEY (RatingTableID, TableKey5)
        REFERENCES EBRatingTableKey(RatingTableID, TableKey),
)
```

3.2.1.2.25 CREATE TABLE EBArea

```
(   AreaNo          VARCHAR(16)     NOT NULL,
    ParentAreaNo    VARCHAR(16)     NOT NULL,
    AreaID          VARCHAR(16)     NOT NULL,
    AreaType        VARCHAR(16)     NOT NULL,
    AreaName        VARCHAR(16)     NOT NULL,

    CONSTRAINT PKArea
        PRIMARY KEY CLUSTERED (AreaNo)
)
```

3.2.1.2.26 CREATE TABLE EBRatingHistory

```
(   ConsumerID      VARCHAR(16)     NOT NULL,
    CarrierID       VARCHAR(16)     NOT NULL,
    ProductID       VARCHAR(16)     NOT NULL,
    PolicyNo        SMALLINT        NOT NULL,
    RatingDate      DATETIME        NOT NULL    DEFAULT GETDATE(),
    RatingNo        INTEGER         NOT NULL    IDENTITY,
    Rating          NUMERIC(12,6)   NOT NULL    DEFAULT 0,
    RatingType      VARCHAR(16)     NOT NULL,

    CONSTRAINT PKRatingHistory
        PRIMARY KEY CLUSTERED (ConsumerID, CarrierID, ProductID, PolicyNo, RatingDate),
    CONSTRAINT UNRatingNo
        UNIQUE NONCLUSTERED (RatingNo),
    CONSTRAINT FKConsumerIDofRatingHistory
        FOREIGN KEY (ConsumerID)
        REFERENCES EBConsumer(ConsumerID),
)
```

3.2.1.2.27 CREATE TABLE EBRatingHistoryDetail

```
(   RatingNo        INTEGER         NOT NULL,
    SegmentNo       INTEGER         NOT NULL,
    SegmentRating   NUMERIC(12,6)   NOT NULL    DEFAULT 0,

    CONSTRAINT PKRatingHistoryDetail
        PRIMARY KEY CLUSTERED (RatingNo, SegmentNo),

    CONSTRAINT FKSegmentofRatingHistoryDetail
        FOREIGN KEY (SegmentNo)
        REFERENCES EBSegment(SegmentNo)
)
```

4. APPENDICES

4.1 EXAMPLE RATE CALCULATION

The mythical Ex InsCo Prism Group Employee Medical Benefits Product

4.1.1 PRODUCT ELIGIBILITY RULES

Group size minimum = 1
Group size maximum = 50
Maternity is mandatory for groups of 10 or more employees.

4.1.2 CONSUMER RISK PROFILE

These parameters were chosen to require the rating engine to use a variety of table entries.

State of Colorado, City of Boulder
Group Size of 4 employees

	Employee	Spouse	#Children
1	30M	25F	2
2	45M		
3	35F	38M	4
4	28F		

Month of rating is May, 1997

4.1.3 POLICY OPTION

These options were chosen to force the most complex calculations for each choice.

Deductible	$750
Coverage %	80%
Stop Loss	$2500
Maternity	Yes
Supplemental Accident & DXL	Yes
Alcohol Rider	Yes (no impact)
Managed Care Factor	PHN with Utilization Review Sloans Lake
Out of Network Differential	30%
PCS Option	Yes

Don't do non-medical coverages in this example.

4.1.4 CARRIER'S ALGORITHM

1. From Medical Base Rates, Rate Basis Type = Maternity (optional): for each employee
   1.1. $143.95
   1.2. $ 56.54
   1.3. $143.55
   1.4. $ 40.30
   1.5. Total of $384.34 $384.34
2. Product Variation Factor
   2.1. Coverage % = 80%
   2.2. Stop Loss = $2500
   2.3. Deductible = $750
   2.4. Factor = 0.6256 $384.34 * 0.6256 = $240.4431
3. Supplemental Accident & DXL: for each employee
   3.1. $6.42
   3.2. $1.63
   3.3. $6.42
   3.4. $1.63
   3.5. Total of $16.10 $16.10 + $240.4431 = $256.5431
4. Rate Area Factor
   4.1. Rate Area = 11
   4.2. Factor = 1.048
   4.3. Exponentiate 1.048 ** (11 - 1) = 1.5981 1.5981 * $256.5431 = $409.9815
5. Managed Care Factor
   5.1. Grouping of 2 0.9184 0.9184 * $409.9815 = $376.527
   5.2. Out of Net Differential 0.93605 0.93605 * $376.527 = $352.4481
6. PCS Option: for each employee
   6.1. $12.57
   6.2. $ 6.81
   6.3. $12.57
   6.4. $ 2.99
   6.5. Total of $34.94 $34.94 + $352.4481 = $387.3881
7. Trend Factor
   7.1. Coefficient is 3.0544
   7.2. Months is 7
   7.3. Monthly trend factor is 1.0125
   7.4. 3.0544 * (1.0125 ** 7) = 3.3320 3.3320 * $387.3881 = $1290.7771
8. Non-medical coverages not considered in this example.
9. The final rating returned is **$1290.78**, which is the monthly premium for the four employees and their dependents.

4.1.5 INSWEB RATING ARRAY SETUP

For each segment and factor the table name, then the returned factor's value for the example is given.
(P) designates a product related factor
(C) designates a consumer related factor

Factor	Segment 1 - Base	Segment 2 - Acc & DXL	Segment 3 - PCS
Starting Base	1.0	1.0	1.0
Age/Sex/Family (C)	MBR - $384.34		PCS - $34.94
Product Variation (P)	PVF - 0.6256		
Family (C)+(P)		SADXL - $16.10	
Area (C)	RAF - 1.5981	RAF - 1.5981	
Managed Care Factor (PC)	MCF (B-NO) - 0.9184	MCF (B-NO) - 0.9184	
Out of Net Differential (P)	DF - .93605	DF - .93605	
Trend (C)	TF - 3.3320	TF - 3.3320	TF - 3.3320
Segment Result	$1100.64	$73.70	$116.42
Rating = $1290.76			

4.1.6 EGBRE CALCULATIONS

Rating = 0

4.1.6.1 Segment One

SegmentRating = Segment.BaseValue = 1
Segment.SeparateYN = "Y"

4.1.6.1.1 Product Factor 1 - Age / Sex / Family

ProductFactor.Algorithm = "T"able
ProductFactor.RatingTableID = MBR
ProductFactor.BaseValue = 1
RatingTable.Dimension = 3
RatingTable.ExpirationDate > RatingDate passed

RatingTableKey.FactorType(1) = "E"mployee
RatingTableKey.TableKey(1) = Age
RatingTableKey.KeyType(1) = "R"ange

RatingTableKey.FactorType(2) = "E"mployee
RatingTableKey.TableKey(2) = Family
RatingTableKey.KeyType(2) = "E"quality

RatingTableKey.FactorType(3) = "C"onsumer
RatingTableKey.TableKey(3) = Maternity
RatingTableKey.KeyType(3) = "E"quality

Because there is a RatingTableKey.FactorType = "E"mployee, a table lookup will be done for each employee and the returned values will be added for all employees.
For each employee in the product's group

Dim3.TableKey1 = Min value >= EmployeeFactor.FactorValue for FactorID = Age {30, 45, 35, 28}
Because RatingTableKey.KeyType(1) = "R"ange, look in table Dim3 for the least TableKey1 greater than EmployeeFactor.FactorValue for FactorID = Age

Dim3.TableKey2 = EmployeeFactor.FactorValue for FactorID = Family {2A+C, 1A, 2A+C, 1A}
Because RatingTableKey.KeyType(2) = "E"quality, look in table Dim3 for the TableKey2 equal to EmployeeFactor.FactorValue for FactorID = Family

Dim3.TableKey3 = ConsumerFactor.FactorValue for FactorID = Maternity {Yes - optional}
Because RatingTableKey.KeyType(3) = "E"quality, look in table Dim3 for the TableKey3 equal to ConsumerFactor.FactorValue for FactorID = Maternity

Add the table values as they are collected. {143.95 + 56.54 + 143.55 + 40.30}

SegmentFactorRating = 384.34
SegmentRating = 1 * 384.34 = 384.34

4.1.6.1.2 Product Factor 2 - Plan Variation

ProductFactor.Algorithm = "T"able
ProductFactor.RatingTableID = PVF
ProductFactor.BaseValue = 1
RatingTable.Dimension = 3
RatingTable.ExpirationDate > RatingDate passed

RatingTableKey.FactorType(1) = "P"lan
RatingTableKey.TableKey(1) = CoveragePercentage
RatingTableKey.KeyType(1) = "E"quality

RatingTableKey.FactorType(2) = "P"lan
RatingTableKey.TableKey(2) = StopLoss
RatingTableKey.KeyType(2) = "E"quality

RatingTableKey.FactorType(3) = "P"lan
RatingTableKey.TableKey(3) = Deductible
RatingTableKey.KeyType(3) = "E"quality

Because there is a RatingTableKey.FactorType = "P"lan one of several possible values can be used for PolicyOption.OptionValue. If there is a ConsumerOption.OptionValue for each coverage then set PolicyOption.OptionValue = ConsumerOption.OptionValue. If not then use the policy's PolicyOption.OptionValue. If there is none, set PolicyOption.OptionValue = Option.OptionValue where DefaultYN = "Y".

Dim3.TableKey1 = PolicyOption.OptionValue for CoverageID = " CoveragePercentage " {80}
Dim3.TableKey2 = PolicyOption.OptionValue for CoverageID = " StopLoss " {2500}
Dim3.TableKey3 = PolicyOption.OptionValue for CoverageID = " Deductible " {750}
Because all 3 RatingTableKey.KeyType = "E"quality, look in table Dim3 for the TableKey equal to the respective values of EmployeeFactor.FactorValue to obtain the value {0.6256}

SegmentFactorRating = 0.6256
SegmentRating = 384.34 * 0.6256 = 240.443104

4.1.6.1.3 Product Factor 3 - Area

ProductFactor.Algorithm = "T"
ProductFactor.RatingTableID = RateAreaFactor
ProductFactor.BaseValue = 1
RatingTable.Dimension = 2
RatingTable.ExpirationDate > RatingDate passed

RatingTableKey.FactorType(1) = "P"lan
RatingTableKey.TableKey(1) = CoveragePercentage
RatingTableKey.KeyType(1) = "E"quality

RatingTableKey.FactorType(2) = "P"lan
RatingTableKey.TableKey(2) = Deductible
RatingTableKey.KeyType(2) = "E"quality

Because there is a RatingTableKey.FactorType = "P"lan one of several possible values can be used for PolicyOption.OptionValue. If there is a ConsumerOption.OptionValue for each coverage then set PolicyOption.OptionValue = ConsumerOption.OptionValue. If not then use the policy's PolicyOption.OptionValue. If there is none, set PolicyOption.OptionValue = Option.OptionValue where DefaultYN = "Y".

Dim2.TableKey1 = PolicyOption.OptionValue for CoverageID = " CoveragePercentage " {80}
Dim2.TableKey2 = PolicyOption.OptionValue for CoverageID = " Deductible " {750}
Look in table Dim2 using the above keys (equality) to obtain the value {1.048}
Because RatingTable.NextRatingTable = CountiesinColorado a second table lookup is done
RatingTable.Dimension = 1
RatingTable.ExpirationDate > RatingDate passed
RatingTableKey.FactorType = "C"onsumer
RatingTableKey.TableKey = Location

Dim1.TableKey1 matches ConsumerFactor.FactorValue for FactorID = Location {Boulder}
Because RatingTableKey.KeyType = "L"ocation, look in table Dim1 for the greatest TableKey1 which is a prefix for ConsumerFactor.FactorValue for FactorID = Location {10}

* Note that design of the area lookup function depends on more detailed requirements to be developed as part of the consumer inquiry interface specification.

Because RatingTable.NextRelationship = "E"xponentiate calculate Dim2.RatingValue ** Dim1.RatingValue {1.048 ** 10 = 1.5981}

SegmentFactorRating = 1.5981327
SegmentRating = 1.5981327 * 240.443104 = 384.25999

4.1.6.1.4 Product Factor 4 - Managed Care Factor (and Out of Net Differential)

ProductFactor.Algorithm = "T"
ProductFactor.RatingTableID = ManagedCareGroupings
ProductFactor.BaseValue = 1
RatingTable.Dimension = 3
RatingTable.ExpirationDate > RatingDate passed

RatingTableKey.FactorType(1) = "P"lan
RatingTableKey.TableKey(1) = Network
RatingTableKey.KeyType(1) = "E"quality

RatingTableKey.FactorType(2) = "P"lan
RatingTableKey.TableKey(2) = ProductType
RatingTableKey.KeyType(2) = "E"quality

RatingTableKey.FactorType(3) = "C"onsumer
RatingTableKey.TableKey(3) = Location
RatingTableKey.KeyType(3) = "E"quality

Because there is a RatingTableKey.FactorType = "P"lan one of several possible values can be used for PolicyOption.OptionValue. If there is a ConsumerOption.OptionValue for each coverage then set PolicyOption.OptionValue = ConsumerOption.OptionValue. If not then use the policy's PolicyOption.OptionValue. If there is none, set PolicyOption.OptionValue = Option.OptionValue where DefaultYN = "Y".

Dim2.TableKey1 = PolicyOption.OptionValue for CoverageID =Network {Sloans Lake}
Dim2.TableKey2 = PolicyOption.OptionValue for CoverageID =ProductType {PHN}
Dim2.TableKey3 = ConsumerFactor.FactorValue for FactorID = Location {Boulder}

Look in table Dim3 using the above keys (equality) to obtain the value {2}
Because RatingTable.NextRatingTable = ManagedCareFactor a second table lookup is done.
Because RatingTable.NextRelationship = "K"ey the value just retrieved (2) is used as a key value in the next table lookup.

RatingTable.Dimension = 5
RatingTable.ExpirationDate > RatingDate passed

RatingTableKey.FactorType(1) = "R"ating Table
RatingTableKey.TableKey(1) = ManagedCareGroupings
RatingTableKey.KeyType(1) = "E"quality

RatingTableKey.FactorType(2) = "P"lan
RatingTableKey.TableKey(2) = UtilizationReview
RatingTableKey.KeyType(2) = "E"quality

RatingTableKey.FactorType(3) = "P"lan
RatingTableKey.TableKey(3) = ProductType
RatingTableKey.KeyType(3) = "E"quality

RatingTableKey.FactorType(4) = "P"lan
RatingTableKey.TableKey(4) = CoveragePercentage
RatingTableKey.KeyType(4) = "E"quality

RatingTableKey.FactorType(5) = "P"lan
RatingTableKey.TableKey(5) = Deductible
RatingTableKey.KeyType(5) = "E"quality

Because there is a RatingTableKey.FactorType = "P"lan one of several possible values can be used for PolicyOption.OptionValue. If there is a ConsumerOption.OptionValue for each coverage then set PolicyOption.OptionValue = ConsumerOption.OptionValue. If not then use the policy's PolicyOption.OptionValue. If there is none, set PolicyOption.OptionValue = Option.OptionValue where DefaultYN = "Y".

Dim5.TableKey1 = value returned from table ManagedCareGroupings {2}
Dim5.TableKey2 = PolicyOption.OptionValue for CoverageID = UtilizationReview {Y}
Dim5.TableKey3 = PolicyOption.OptionValue for CoverageID = ProductType {PHN}
Dim5.TableKey4 = PolicyOption.OptionValue for CoverageID = CoveragePercentage {80}
Dim5.TableKey5 = PolicyOption.OptionValue for CoverageID = Deductible {750}

Look in table Dim5 using the above keys (equality) to obtain the value {0.9184}
Because RatingTable.NextRatingTable = DifferentialFactors, yet a third table lookup is done.
Because RatingTable.NextRelationship = "M"ultiply the value just retrieved (2) is multiplied by the value to be retrieved in the next table lookup.

RatingTable.Dimension = 2
RatingTable.ExpirationDate > RatingDate passed

RatingTableKey.FactorType(1) = "P"lan
RatingTableKey.TableKey(1) = ProductType
RatingTableKey.KeyType(1) = "E"quality

RatingTableKey.FactorType(2) = "P"lan
RatingTableKey.TableKey(2) = OONetDifferential
RatingTableKey.KeyType(2) = "E"quality

Because there is a RatingTableKey.FactorType = "P"lan one of several possible values can be used for PolicyOption.OptionValue. If there is a ConsumerOption.OptionValue for each coverage then set PolicyOption.OptionValue = ConsumerOption.OptionValue. If not then use the policy's PolicyOption.OptionValue. If there is none, set PolicyOption.OptionValue = Option.OptionValue where DefaultYN = "Y".

Dim2.TableKey1 = PolicyOption.OptionValue for CoverageID = ProductType {PHN}
Dim2.TableKey2 = PolicyOption.OptionValue for CoverageID = OONetDifferential {30}
Look in table Dim3 using the above keys (equality) to obtain the value {0.93605}

Because RatingTable.NextRelationship = "M"ultiply calculate Dim5.RatingValue * Dim2.RatingValue {0.9184 * 0.93605 = 0.85966832}

SegmentFactorRating = 0.85966832
SegmentRating = 384.25999 * 0.85966832 = 330.33614

4.1.6.1.5 Product Factor 5 - Trend

ProductFactor.Algorithm = "T"able
ProductFactor.RatingTableID = Trend
ProductFactor.BaseValue = 3.0544
RatingTable.Dimension = 0
RatingTable.ExpirationDate > RatingDate passed

RatingTableKey.FactorType(1) = "C"onsumer
RatingTableKey.TableKey(1) = RatingDate
RatingTableKey.KeyType(1) = "T"rend

Look in table Dim0 using the above keys (equality) to obtain the RatingValue {1.0125}
Because the Dimension is 0, no key is actually necessary for the table lookup (but it is required to locate the KeyType!). Because the KeyType is "T", the RatingValue is raised to a power equal to the number of months between the RatingDate and the Product.TrendDate{7}. {1.0125 ** 7}

SegmentFactorRating = 3.0544 * 1.0908505 = 3.3318937
SegmentRating =330.33614 * 3.3318937 = 1100.6449

4.1.6.1.6 Segment One Result

The resulting segment one rating is $1100.64

4.1.6.2 Segment Two

4.1.6.2.1 Product Factor 1 - Family (Supplemental Accident)

ProductFactor.Algorithm = "T"able
ProductFactor.RatingTableID = SADXL
ProductFactor.BaseValue = 1
RatingTable.Dimension = 3
RatingTable.ExpirationDate > RatingDate passed

RatingTableKey.FactorType(1) = "E"mployee
RatingTableKey.TableKey(1) = Family
RatingTableKey.KeyType(1) = "E"quality

RatingTableKey.FactorType(1) = "P"lan
RatingTableKey.TableKey(1) = CoveragePercentage
RatingTableKey.KeyType(1) = "E"quality

RatingTableKey.FactorType(2) = "P"lan
RatingTableKey.TableKey(2) = Deductible
RatingTableKey.KeyType(2) = "E"quality

Because there is a RatingTableKey.FactorType = "P"lan one of several possible values can be used for PolicyOption.OptionValue. If there is a ConsumerOption.OptionValue for each coverage then set PolicyOption.OptionValue = ConsumerOption.OptionValue. If not then use the policy's PolicyOption.OptionValue. If there is none, set PolicyOption.OptionValue = Option.OptionValue where DefaultYN = "Y".

Because there is a RatingTableKey.FactorType = "E"mployee, a table lookup will be done for each employee and the returned values will be added for all employees.
For each employee in the product's group
Dim3.TableKey1 = EmployeeFactor.FactorValue for FactorID = Family {2A+C, 1A, 2A+C, 1A}
Because RatingTableKey.KeyType(1) = "E"quality, look in table Dim3 for the TableKey1 equal to EmployeeFactor.FactorValue for FactorID = Family

Dim3.TableKey2 = PolicyOption.OptionValue for CoverageID = CoveragePercentage {80}
Dim3.TableKey3 = PolicyOption.OptionValue for CoverageID = Deductible {750}

Add the table values as they are collected. {6.42 + 1.63 + 6.42 + 1.63}

SegmentFactorRating = 16.1
SegmentRating = 1 * 16.1 = 16.1

4.1.6.2.2 Product Factor 2 - Area

This is the same as Segment 1, Factor 3

SegmentFactorRating = 1.5981327
SegmentRating = 1.5981327 * 16.1 = 25.729936

4.1.6.2.3 Product Factor 3 - Managed Care Factor (and Out of Net Differential)

This is the same as Segment 1, Factor 4

SegmentFactorRating = 0.85966832
SegmentRating = 25.729936 * 0.85966832 = 22.119211

4.1.6.2.4 Product Factor 4 - Trend

This is the same as Segment 1, Factor 5

SegmentFactorRating = 3.3318937
SegmentRating = 22.119211 * 3.3318937 = 73.698861

4.1.6.2.5 Segment Two Result

The resulting segment two rating is $73.70

4.1.6.3 Segment Three

4.1.6.3.1 Product Factor 1 - Age/Sex/Family

ProductFactor.Algorithm = "T"able
ProductFactor.RatingTableID = PCS
ProductFactor.BaseValue = 1
RatingTable.Dimension = 2
RatingTable.ExpirationDate > RatingDate passed

RatingTableKey.FactorType(1) = "E"mployee
RatingTableKey.TableKey(1) = Age
RatingTableKey.KeyType(1) = "R"ange

RatingTableKey.FactorType(2) = "E"mployee
RatingTableKey.TableKey(2) = Family
RatingTableKey.KeyType(2) = "E"quality

Because there is a RatingTableKey.FactorType = "E"mployee, a table lookup will be done for each employee and the returned values will be added for all employees.
For each employee in the product's group

Dim2.TableKey1 = Min value >= EmployeeFactor.FactorValue for FactorID = Age {30, 45, 35, 28}
Because RatingTableKey.KeyType(1) = "R"ange, look in table Dim2 for the least TableKey1 greater than EmployeeFactor.FactorValue for FactorID = Age

Dim2.TableKey2 = EmployeeFactor.FactorValue for FactorID = Family {2A+C, 1A, 2A+C, 1A}
Because RatingTableKey.KeyType(2) = "E"quality, look in table Dim2 for the TableKey2 equal to EmployeeFactor.FactorValue for FactorID = Family

Add the table values as they are collected. {12.57 + 6.81 + 12.57 + 2.99}

SegmentFactorRating = 34.94
SegmentRating = 1 * 34.94 = 34.94

4.1.6.3.2 Product Factor 2 - Trend

This is the same as Segment 1, Factor 5

SegmentFactorRating = 3.3318937
SegmentRating = 34.94 * 3.3318937 = 116.41636

4.1.6.3.3 Segment Three Result

The resulting segment two rating is $116.42

4.1.6.4 Final Rating

The resulting rating is $1290.76